CONCEPTUAL DESIGN OF AN ATTACHMENT BASED RECONFIGURABLE MACHINE TOOL USING DESIGN STRUCTURE MATRIX

by

Ambrish Gupta

B.Technology, R.E.C. Jalandhar, India

Toronto, Canada, 2004

A thesis

presented to **Ryerson University**

in partial fulfillment of the requirement for the degree of

Masters of Applied Science

in the program of

Mechanical Engineering

Toronto, Ontario, Canada, 2004

Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Borrower's Page

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

CONCEPTUAL DESIGN OF AN ATTACHMENT BASED RECONFIGURABLE MACHINE TOOL USING DESIGN STRUCTURE MATRIX

MASc 2004, Ambrish Gupta, Mechanical Engineering, Ryerson University, Toronto.

There has been very little research in the field of Reconfigurable machine Tools (RMTs). None of the past research developed a method to design a set of RMT configurations required to machine a part-family.

This thesis presents a novel method to determine the functional specifications of the RMT configurations required to machine a part-family. The method is developed by firstly designing the RMT required to machine a single part. Thereafter, this method is extrapolated to suit the problem of a part-family.

To design the RMTs for a single part, firstly, the part is decomposed into manufacturing features. Next, a novel method is developed to cluster the features. Each of these feature-clusters corresponds to a single RMT configuration. Based on the machining requirements of these RMT configurations, the modules are designed. These modules are assembled to form the final RMTs. The method is demonstrated by applying it to an example part.

Keywords: Reconfigurable Machine Tool, Design Structure Matrix.

Acknowledgements

Firstly, I would like to quote the following *Shloka* (*verse*) from '**Rig Veda**' (the oldest Hindu Scripture):

Sanskrit: **"Prano devi saraswathi vajebhir vajinavathi dhinamavidrayavathu"**

English Translation: "Salutations to Mother Saraswathi (the Goddess of Knowledge) who is the creator of all vibrations and who removes the ignorance of the mind".

I want to quote the following Shloka (*verse*) from 'Rig Veda':

Sanskrit: **"Gurur brahmaa gurur vishnuh, gurur devo maheshvarah, gurur saakshaat parabrahma, tasmai shree gurave namah."**

English Translation: "Know the Guru (*Teacher*) to be Brahma (*The Creator*) himself. He is Vishnu (*The Preserver*). He is also Shiva (*The Destroyer*). Know Him to be the Supreme Brahman (*the Supreme Transcendental God*), and offer thy adorations unto that peerless Guru."

I was privileged to have worked under the supervision of Dr.F.Salustri and Dr.F.Xi. They are "Alchemists" in true sense.

I consider myself as a high enthalpy, high entropy system and at times I was tempted to go in many wrong directions. These superior mortals steered my 'free energy' without suffocating my creative and wandering spirit. I sincerely appreciate their omniscience, foresight, and compassion.

I would also like to express gratitude to Dr.F.Sharifi for laying my foundations in the field of robotics. He kindled my interest in the field and gave me a vision. All my work, this and future, dwell on the foundations he laid.

The first person I would like to thank on a personal note is my Godfather, Shri.S.Chand, who nurtured me and gave dimensions to my thought. He was a teacher, and friend and a perennial source of strength. His spirit and teachings always guide me. I credit all my achievements to him.

A quote once again, **"Feelings of worth can flourish only in an atmosphere where individual differences are appreciated, mistakes are tolerated, communication is**

open, and rules are flexible- the kind of atmosphere that is found in a nurturing family" -Virginia Satir

I was blessed to have a supporting family. Without their multi-dimensional support, I would never have accomplished anything substantial. I especially thank my parents for being a constant source of inspiration.

I would like to express my gratitude to 'Ryerson University' for its SGS scholarship.

Dedication

To my Parents.

Table of Contents

List of Figures

List of Tables

Nomenclature

Table A: Symbols used in this thesis

Symbol	Meaning
$\forall$	For all
$\vee$	Logical OR
$\wedge$	Logical AND
Φ	Null Set
:=	Assignment

Table B: Acronyms used in this thesis

Acronym	Expansion
AD	Axiomatic Design.
AM	Atomic Modules (a motion in a single direction).
API	Application Protocol Interface.
ASIC	Application Specific Integrated Circuit.
Ci	Feature Cluster # i.
CA	Cellular Automata.
CAPP	Computer Aided Process Planning.
CBR	Case Based Reasoning.
F(Q)KC	Flexible (Quasi) Kinematic Coupling.
DOF	Degree Of Freedom.
DM	Design Matrix.
DMS	Dedicated Manufacturing System.
DSM	Design Structure Matrix.
f	Manufacturing Feature.
FMEA	Failure Mode and Effect Analysis.
FMS	Flexible Manufacturing System.
FPGA	Field Programmable Gate Array.
FKC	Flexible Kinematic Coupling
FR	Functional Requirements.
FSM	Finite State Machine
GA	Genetic Algorithm.
MHS	Material Handling System.
OS	Operation Space.
PI	Prime Implicant.
PSM32	DSM tool used in this thesis. <www.problematics.com>
RMS	Reconfigurable Manufacturing System.
RMT	Reconfigurable Machine Tool.
ROI	Return On Investment.
RR	Reconfigurable Robot.
SOM	Self Organising Map.
STEP	STandard for the Exchange of Product Data.
STEP-AP	STEP-Application Protocol
TAD	Tool Approach Direction.
UML	Unified Modeling Language.
VLSI	Very Large Scale Integration.
w.r.t.	With Respect To.
WWW	World Wide Web.

1. Introduction

This chapter firstly introduces the evolution of machining systems. This is followed by a comparison between the different machining systems. Next, the problem-statement, and the scope of the thesis are presented. This is followed by presenting the approach, and finally, the organization of this thesis.

1.1. Background

Increasing the 'Return on Investment' (ROI) has been the paramount objective of the business community. For manufacturing companies, the main investment is in machines. Better returns can be reaped by increasing the production capacity of the machines, or by extending their life. This section explains briefly the evolution of machining systems as an effort to fulfill this objective.

Products developed in the previous decades were characterized by monopolies and having long technological lives. Therefore, 'Dedicated Machining Systems' (DMS) were designed. They were part-centred machines, using fixed transfer lines, and yielding good quality and high production rates. Moreover, their technological lives were also long because of the fixed variety of products to manufacture. Therefore, using DMS was economically promising.

On the other hand, today's dynamic markets are characterized by a huge variety of products and quick obsolescence. This implies that manufacturing enterprises must adapt more quickly and more frequently. Using DMS would be unviable because any change to the task-set would imply a major change to the machining system. Thus, the ROI would fall sharply. To meet this challenge, researchers developed 'Flexible Machining Systems' (FMS). These machining systems were equipped with all the functionality that they may ever have to manifest. Although they met the challenge of being versatile, they do not yield the best ROI. The prime reason being that according to the Pareto Principle, 80% of manufacturing tasks require 20% of functionality. This implies that most of the functionality is highly under-utilized and so is the invested capital. Similar efforts of equipping systems with all the possible functionalities have been observed in many of the current word-processing software. As in the FMS, the majority of the 'features' remains

highly under-utilized. Such an overstuffing of products with functionality is called "Feature Creep," and leads to "Cost Creep." [39, 99]

Considering the benefits and shortcomings of DMS and FMS, it has been concluded that neither of these assures the best ROI for a typical manufacturing company (which is characterized by a limited variation production portfolio). Similar situations have been observed in robotics and computing hardware. This problem was solved there by applying modular reconfigurability to design systems. Brief examples are presented below to give a flavor of this.

1) Exploration robots are terrain specific e.g., rovers are best for plain terrain, crawlers for rugged ones, swimmers for under-water exploration *etc.* Therefore, for a generic exploratory task, no single-function robot is optimal. To meet this challenge, the robot should allow itself to be configured into different task-specific-configurations, which are characterized by a particular set of 'physical' and 'control system' features. The robot configures such that, the resulting configuration best suits the specific terrain/environment. These kinds of robots are called, 'Reconfigurable Robots' (RRs). They are analogous to cold-blooded animals in the sense that the RRs change their topologies and controls based on the external environment, just as these animals change their internal temperature (within a certain range) to suit the environment. For example, TetroBot, a parallel-reconfigurable robot, [23] can configure itself into a crawler, a meteorological station, and a drilling station.

   Space transportation is very expensive (at least until the space elevator [2] is made), and RRs reduce the weight to be transported because many application specific robots can be reduced to a fewer number of RRs.

2) Robots participating in RoboCup[1] are typically required to change their behavior in accordance with the situation. Although using microprocessor-based control hardware fulfills this requirement, they do not exhibit real-time response. On the other hand, an 'Application Specific Integrated Circuit' (ASIC)-based controller guarantees real-time response but does not offer flexibility. Therefore, the ideal hardware is neither

---

[1] Ref: <www.robocup.org>

flexible, nor dedicated, but 'Reconfigurable'. Such hardware would allow the 'brain' of the robot to assume different behaviors e.g., center-forward, defenseman *etc.* Another benefit of using reconfigurable hardware is the possibility of learning while in operation.

3) Machining a flat-bottom pocket on a metal slab, requires a 2.5 DOF vertical milling machine. For another task of machining a pocket, on the flat surface a long shaft, a 2.5 DOF horizontal milling machine is needed. Therefore, a manufacturer producing these two parts would require two dedicated machines, viz., 2.5 DOF horizontal milling machine, and a 2.5 DOF vertical milling machine. A 5 DOF flexible machine could replace these two independent machines. The shortcoming of using a flexible machine is that for machining either part, the machine is utilizing only half of its functionality. To solve this problem, reconfigurable machines were developed. A reconfigurable machine can be configured into the dedicated configurations of 2.5 horizontal/vertical milling machine. Such a machine is flexible to be dedicated. This characteristic leads to both high utilization and good performance.

The successful development of RRs led researchers to apply modular reconfigurability to machining systems. This paradigm was called 'Reconfigurable Machining System' (RMS).

RMS is defined as a rapid-response machining system characterized by convertibility and customization to meet changes in the production-portfolio. It can respond to changes in the production-portfolio by changing its topology as well as its controls. In other words, the RMS is 'flexible to be dedicated', i.e., it can be 'configured' to have part(s)-specific functionality, and once configured, it yields a performance comparable to that of a DMS. These characteristics of the RMS are attributable to modularity of the system and its components to varying degrees of granularity (level of modularization). Moreover, modularity also helps easy updating of the system components. Thus, RMSs are not vulnerable to obsolescence and enjoy a potentially infinite life span.

Unlike FMS, RMS is typically designed around a part-family. The different configurations of the RMS correspond to a subset of the tasks posed by the machining requirements of the part-family. Although a RMS is designed around a part-family, it

offers a fair level of re-use of system components if a new part extraneous to the part-family, around which the RMS is designed, arrives. The degree of re-use generally depends on the granularity of the modularization of the RMS components.

As shown in Figure 1.1, and Table 1-1, RMS embodies the advantages of both DMS, and FMS; thus, making RMSs technologically and economically viable.

The main machining component of a RMS is the 'Reconfigurable Machine Tool' (RMT). A RMT is a machine tool characterized by modular construction. It can reconfigure in terms of construction as well as control strategy to assume roles of different dedicated machines. The RMT is configured for a specific task-set in order to manifest the tasks by having minimum redundancy.

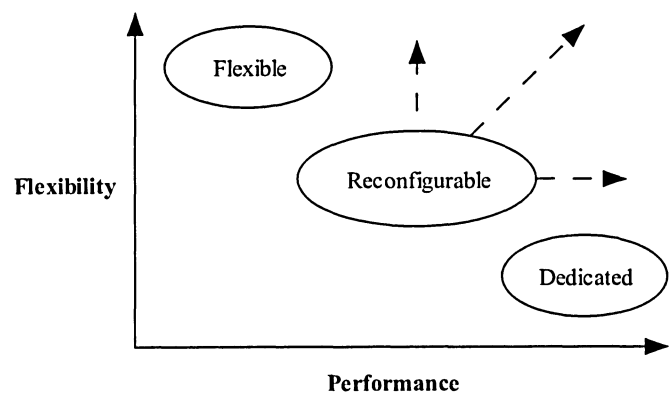


Figure 1.1: Performance vs. Flexibility of Machining Systems (Koren and Ulsoy [41])

Reconfigurable systems must satisfy the essential requirement of being modular. These modules are functionally independent components of the system. When assembled in a configuration, they cooperate to manifest the assigned tasks. Drawing an analogy to swarm intelligence exhibited by insects, each module is equivalent to an ant and the system configurations to the ant-colony. The scope of this thesis is limited to defining the modules in terms of the motions they provide. Therefore, in this thesis, the modules that form the RMTs are referred to as motion-modules.

The concept of reconfigurability is an extension of the concept of modularity from the domain of design and manufacturing to the domain on operation. The basic goal of modularity is to control complexity, improve manufacturability, lower cost *etc.* The goal

of reconfigurability is to have multiple functional-modes, i.e., be able to exhibit multiple configurations, each of which is tailored to a particular set of tasks.

For example, Canada-Arm2 is modular. It is formed by assembling 'Orbital Replacement Units' [8, 49]. However, it does not belong to the category of 'reconfigurable robots' because it exhibits only a single functional mode. On the other hand, TetroBot allows numerous configurations, each formed by assembling tetrahedral/octahedral modules.

RMTs are being accepted by industry. "*Tri-Way Manufacturing Technologies Corp* believes that employing RMS gave it a competitive advantage. *Olympia machine tools,* design RMTs, which are making their way into mainstream manufacturing." [14]

Table 1-1: Comparison of DMS, RMS and FMS [41]

	Dedicated Machining System	Reconfigurable Machining System	Flexible Machining System
Definition	A machining system using transfer line technology with fixed tooling and automation to cost-effectively produce one specific part at high volumes and at the required quality.	A machining system characterized by modular hardware and software. It can cost-effectively manufacture part-families, at the required volume and quality.	A machining system using a fixed hardware and fixed, but programmable software to cost-effectively manufacture a wide variety of parts in small volumes, at a medium quality grade.
Benefits	High productivity & quality. Best for a part with a very long life	Best ROI. Easy troubleshooting, updating. Less susceptible to obsolesence.	Can machine all manufacturable parts that require motions within the machine's workspace.
Shortcomings	If the part to be produced changes, the machining system itself would require a major change to it; in some cases, a complete replacement. Therefore, it yields a very poor ROI for a company with a 'dynamic' production-portfolio.	High upfront cost. Therefore, break-even is achieved late.	Medium productivity & quality. Poor ROI because most of the products require very little of the available functionality.
Use w.r.t. Process planning	Based on the available machines, the part's features are grouped into setups. These setups are then machined by the dedicated machines. The part travels between these machines, on a fixed transfer line.	Machines configured for the part or a set of its features.	There is no choosing of the machine. All the different parts can be machine one the machine. The machine is programmed for different parts.
Notes	There exist 'attachments' to render DMS components for potential re-use (in case, the part to be produced changes). However, a DMS with attachments is not 'Dedicated', because the attachments are 'generic'.Moreover, appending attachments usually adds redundancy; a typical example would be to add a 'Live-spindle' to a conventional-milling machine.		Although, the Flexible machine has all the (perceivable) functionality, yet it possibly cannot machine the complete part in a single setup. This is because the fixturing does not allow approach to all the faces. Therefore, in this case too, the manufacturing features (of the part) are extracted and grouped into setups.

1.2 Problem Statement

As shown in Fig.1.2, the problem addressed in this thesis is to design a set of RMT configurations that can machine a given part-family. This design task consists of designing the motion-modules and configuring the RMTs.

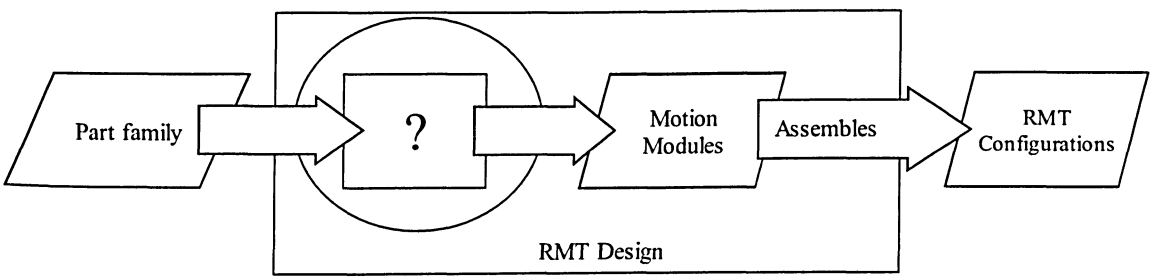


Figure 1.2: The main research objective

1.3. Scope

This thesis focuses on giving a functional description of the motion-modules and the RMTs configurations. The functional description is with respect to motions only. A design method is laid out, supported with a general, non-detailed design of the physical-modules embodying all the assigned functions. The step to perform a detailed design of modules and RMTs is beyond the scope of this thesis.

The research documented in this thesis belongs to the 'proof of concept' category. The method developed here lays out 'black box' tasks to design RMTs. Each of these black boxes implements a method to fulfill a particular task. To show the feasibility of the method, a reasonable method was chosen for each black box. The methods have been chosen such that the input and output formats are STEP (STandard for Exchange of Product data) compliant. However, no particular STEP-Application Protocol was implemented.

These black boxes could feasibly implement any other method without affecting the methods used in other black boxes. This research does not make any commitment to the optimality of the methods chosen for any particular task.

The motion-modules determined by this method could be modularized further, i.e., at a lower level of modularization. This would add more versatility to the system, but increase the cost.

1.4. Design Approach

As shown in Figure 1.3, this thesis firstly scales down the problem of designing a set of RMT configurations for a part-family to that of designing the RMT configurations for a single part. Having developed a solution for the scaled-down problem, the solution is scaled-up to solve the original problem.

Starting with the process for a single part allowed the author to focus on the development and explanation of the process itself, unburdened by the complexity and detail of treating a whole part-family. Scaling the single-part process up to treat a part-family is then done to show the feasibility of the process in 'real' cases.

Reconfigurable systems must be designed with due consideration to the large scale system in which they will be used, i.e., they must be designed in response to the expected purpose of the whole reconfigurable machine. Using 'bottom-up' design strategy does not design with respect to the overall purpose of the machine; hence, it cannot reliably result in a good reconfigurable machine. Therefore, 'top-down' strategy is employed.

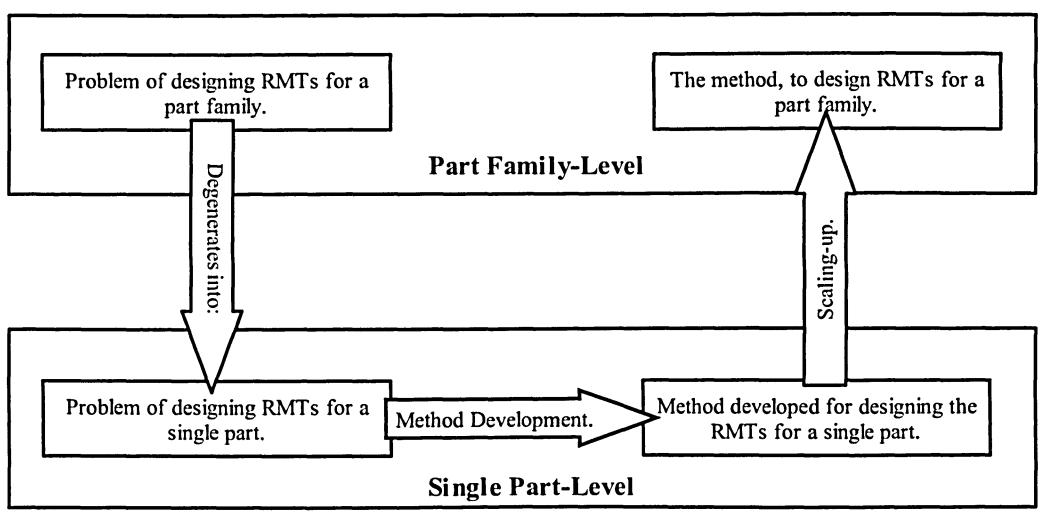


Figure 1.3: The approach for solving the stated problem

1.5. Thesis Outline

Chapter 2 presents the Literature review of the applications of the 'Reconfigurable concept' and 'Setup Planning'. Chapter 3 gives an introduction to the 'Design Structure Matrix' (DSM), which was a key tool used in this thesis. Chapter 4 presents the method for designing the RMT configurations required to machine a part-family.

Results, discussion, and future work are presented in Chapter 5.

This research considers separate modules for those manifesting coordinated motions and sequential motions. Appendix I presents the rationale for the modeling these two types separately.

The appendix is followed by glossary and references.

1.6 Summary

RMS is a new technology developed to suit today's typical manufacturer. RMT is the main machining component of RMS. This thesis focuses on determining the functional specification of the RMTs required to machine a part-family. According to the approach, the original problem is scaled down to determine the specification for the RMTs required to machine a single part. The solution developed for this scaled-down problem is then extended to determine the specification for the RMTs required to machine a part-family.

2. Literature Survey

This chapter surveys the past research in two main domains: (a) Reconfigurable robots, machine tools, computing hardware and fixturing, and (b) 'Setup planning'.

2.1. Survey on Reconfigurable Robots

This section surveys recent research in the field of reconfigurable robots (RRs). Research in this area is investigated with an anticipation of extrapolating pertinent concepts to meet the problem statement.

Hamlin and Sanderson [23] developed 'Tetrobot'. This robot belongs to the class of 'reconfigurable parallel robotic systems'. Tetrobot uses a novel joint mechanism called the, 'Concentric Multi-link Spherical (CMS) joint'. The CMS joint allows an arbitrary number of struts to be connected together and to share a common center of rotation. Thus, Tetrobot can have any statically determinate configuration such as a double octahedral platform, a tetrahedral arm, a six-legged walker *etc.*

PARC [63] developed 'Polypod' and its successor, 'PolyBot'. These robots belong to the class of 'chain reconfigurable robots'. Moreover, they have only one type of module, i.e., the robot is 'unit-modular'. The module has a single Degree Of Freedom (DOF) and has hermaphroditic (genderless) connection plates.

Yim *et al* [108] presented a few interesting space applications of PolyBot. PARC [63] also developed 'Proteo'. The Proteo is a series of simulations of 'Quasi Fluid Crystalloid Robots', using rhombic dodecahedron shaped modules.

Rus and Vona [68] developed the 'Crystalline-robot'. Their robot belongs to the category of 'Lattice Robots'. This robot is a unit modular self-reconfigurable robot. Each unit is a 2-Dimensional square module, and has connectors at the faces. These modules were called, 'crystalline-atoms'. An atom has two centrally placed, orthogonal-prismatic DOFs. These DOFs give 'Atoms' the ability to contract by a factor of two in the X and Y dimensions. The robot reconfigures by a coordinated contraction/expansion of 'Atoms'.

Butler *et al* [7] developed a Cellular automata based reconfiguration planning for their crystalline robot. Their research was an extension of Vona's [86] 'melt and grow

algorithm'. Suh *et al* [77] developed 'TeleCubes', which was an extension of 'crystalline robot' into the third-dimension.

Unsal [82] developed the 'I-Cubes', a self reconfigurable robot. This robot's configurations are formed by assembling two types of modules *viz.* 'links', and 'cubes'. 'Links' are the active elements. They are embedded with 3-DOF mechanisms. 'Cubes' are the passive elements and act as connectors. "Using actuation and attachment properties of 'links' and 'cubes', the system can self- reconfigure" [30]. In another work, Unsal and Khosla [81] presented the mechatronic design of these robots.

Conro [9, 80] is a unit-modular self-reconfigurable robot. Each module has 2-DOFs. Instead of having hermaphroditic connection plates, Conro's modules have three male connectors at one end, and three female connectors at the other. The modules use a 'Shape Memory Alloy' based locking mechanism to connect the modules. "Such a design would easily form tree-like structures (like those of limbed animals) as well as structures with single loops, but none with more than one loop" [30].

Michael [51] developed "Fractal robot," a unit-modular polymorphic robot. The module has screw and groove mechanisms at each cubic face. This allows the robot to perform the assigned tasks as well as reconfigure itself.

None of the afore-mentioned researches followed a systematic method to design the modules. Xi *et al* [100] developed a systematic method for designing modules. They developed a parallel reconfigurable robot for space applications. The modules were designed with an objective to maximize the number of possible configurations, while minimizing the number of modules.

Yang and Chen [101] gave a systematic method for designing the robot configurations. Configuration design determines the topology, which best fulfils the assigned tasks. "A lower number of DOFs assures relatively simpler configurations, higher load carrying capacity, and low power consumption" [101]. Considering this fact, they aimed to generate a task-based-robot with a minimum number of DOFs. Since the search space was discrete, they chose Genetic Algorithm (GA), as the optimization algorithm. The task constraints of reachability, manipulability, and mechanical constructability were integrated into the optimization criteria.

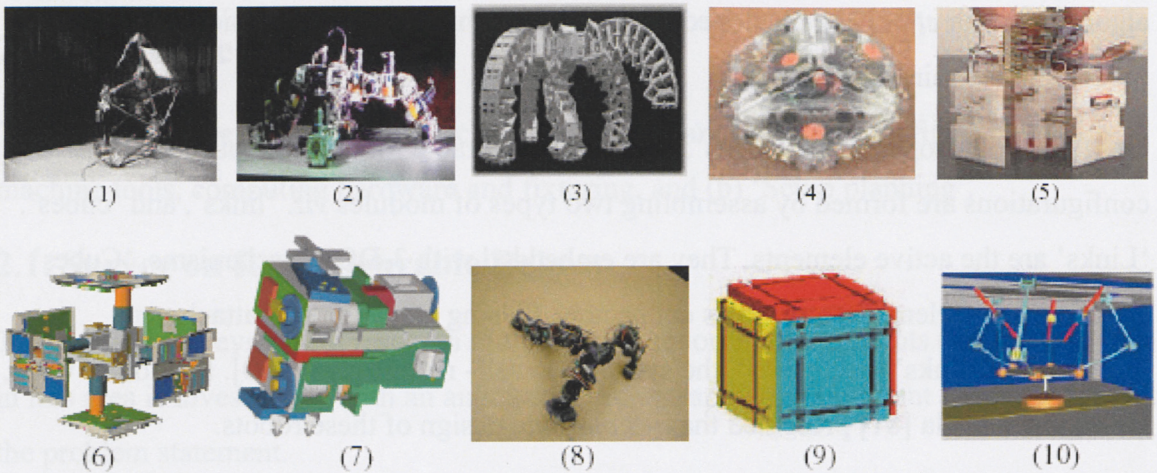


Figure 2.1: Pictures of surveyed reconfigurable robots [23, 63, 68, 82, 51, 100]

Figure 2.1 shows: (1) Octahedral configurations of TetroBot (2) Spider configuration of Polybot (3) 5 legged spider configuration of Polypod (4) Proteo Molecule (5) Crystalline robot Module (6) Telecube Module (7) I-Cubes Module (8) Configuration of CONRO Robot (9) Fractal Robot (10) Reconfigurable parallel robot.

In Table 2-1, 'Y', and 'N' correspond to 'Yes', and 'No' respectively. No data was found regarding the "load carrying capacity" of RRs except for Tetrobot [23]. Therefore, the different RRs could not be compared in this aspect.

Table 2-1: Comparison of the surveyed Reconfigurable-Robots [30]

	Number of DOFs per Unit	Unit-Modulae	3D	Self Reconfiguring
Tetrobot	3 to 5	Y	Y	N
Polypod	2	N	Y	N
PolyBot	1	Y	Y	Y
Proteo	0	Y	Y	Y
Crystalline	2	Y	N	Y
TeleCubes	3	Y	Y	Y
I-Cubes	3	N	Y	Y
Conro	2	Y	Y	Y
Fractal Robot	6	Y	Y	Y
Parallel Reconfigurable Robot(Xi *et al* [100])	2 to 5	N	Y	Y

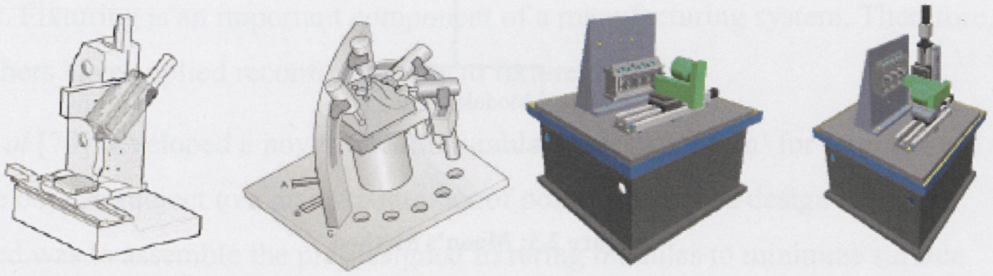
Since, the main application for RRs has been in extra-terrestrial explorations, RRs were developed to have an infinite number of useful configurations and be fault-tolerant. Both these objectives require the RRs to be hyper-redundant.

2.2 Survey on Reconfigurable Machine Tools (RMTs)

This section covers recent research in the field of RMT design. These works are surveyed to understand the methodologies used and search for the methods that can be applied or extended to solve the stated problem.

Recall that the objective of a RMT is to be easily convertible and to yield good performance. The applications for RMTs are quite different from those of RRs. RMTs have more stringent stiffness requirements and require only a relatively fewer configurations.

Katz and Moon [35] developed a prototype of a virtual-arch-RMT (Figure 2.2 (i)) and conducted studies on its different configurations. Koren and Kota [40] were the pioneers of the RMT. The RMT, for which they hold a patent, is shown in Figure 2.2(ii). Landers and Min [42] developed a 2, and 3 axis orthogonal RMTs (Figure 2.2 (iii, iv)) for machining V6 and V8 engine's cylinder heads. All these works were novel, yet none of these gave a systematic methodology for RMT design.



(i) (ii) (iii) (iv)

Figure 2.2: Pictures of surveyed RMTs [35, 40, 42]

Moon [55] developed a systematic methodology for the configuration design of RMTs. The RMT configurations were formed by assembling both active and passive modules. Active modules were used to manifest machining and positioning motions. The passive modules acted as connectors or were embedded with actuators, which are actuated only to

manifest reconfiguration. This research was a major step towards the configuration design of RMTs. However, it was also not clear as how his methodology handled sculpted manufacturing features.

Similar to Yang and Chen [101], Moon followed the bottom-up strategy (shown in Figure 2.3). However, Moon's work differed from that of Yang and Chen, in the following aspects:

- Moon chose from a large set of modules, while Yang and Chen had only three types of modules.
- Moon considered stiffness as a major design criterion, while Yang and Chen did not.
- Yang and Chen used GA to find the optimal configuration. On the other hand, Moon evaluated each of the feasible configurations and chose the one that best satisfied the criteria considered.
- Yang and Chen did not consider passive DOFs, while Moon did.

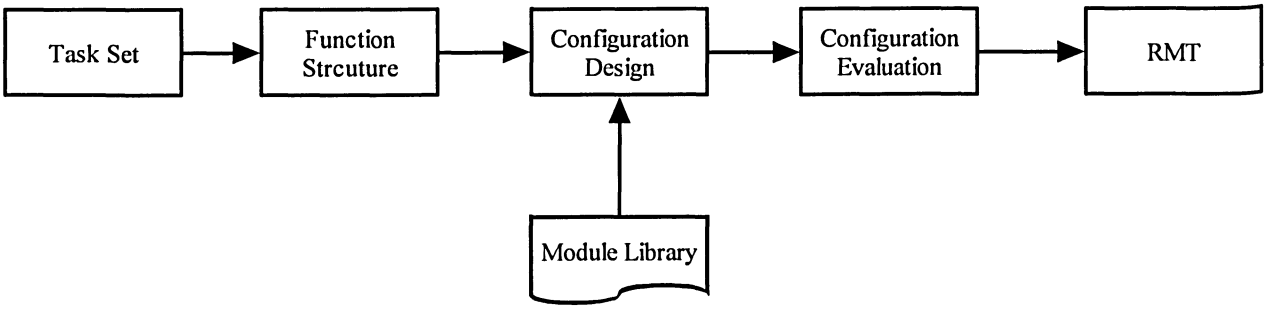


Figure 2.3: Moon's Method

Moon and Kota [57] also developed a 'reconfigurable power spindle', which can machine multiple features simultaneously.

Li *et al* [44] investigated various coupling methods for assembling modules and concluded that using "Flexible kinematic coupling," for connecting modules results in the least errors. However, their research did not consider "Quasi-Kinematic Coupling (QKC)" [12], which can potentially yield a better performance than FKC.

Moon *et al* [54] developed a method for error estimation of RMT configurations. Dabling, and Chase [13] presented a method for tolerance analysis of a machine. In addition to giving a systematic method, they also automated their scheme using MSC.ADAMS. There is a potential of merging Moon *et al*'s scheme with that of Dabling and Chase to analyze the error characteristics of the RMT configurations.

There have also been good research efforts in controls design for RMTs [36, 47, 79] and the economics of RMTs [11, 41], but these are only peripheral to the scope of this thesis.

Based on the survey on RRs and RMTs, it has been observed that the main difference between RRs and RMTs is the design intent. RRs are intended to be hyper-redundant and ideally have an infinite number of configurations, while the design intent for a RMT is to have minimum redundancy and a limited number of configurations. As a corollary RMTs are not fault-tolerant in the way RRs are.

None of the surveyed work gave a method to design both, the modules and the configurations for RMTs required to manufacture a part-family. This thesis aims to address this problem.

2.3 Survey on other applications of the Reconfigurability

If a machining system is to be truly reconfigurable, it should be reconfigurable in all aspects. Fixturing is an important component of a manufacturing system. Therefore, researchers have applied reconfigurability to fixture design.

Sela *et al* [70] developed a novel 'reconfigurable fixturing system' for thin-walled, flexible objects subject to a discrete number of point forces. The design objective followed was to assemble the pre-designed fixturing modules to minimize surface deflections due to external forces.

Shirinizadeh [72] presented a system for computer-aided design and analysis of reconfigurable fixtures. This research uses a set of pre-designed fixturing modules. They developed a software program, which uses the information retrieved from a CAD database to perform kinematic analysis of the fixture configuration. This research did not give a design strategy. Moreover, it required human assistance for designing fixturing configurations.

Reconfigurability has also been applied to the field of 'Computing hardware'. Even though the research most pertinent to RMT design has been surveyed and reported in the previous sections, 'Reconfigurable Computing hardware' is surveyed to gain more insight into reconfigurability and appreciate the breadth of the field.

Reconfigurable computing hardware uses the reconfigurable aspects of 'Field Programmable Gate Arrays' (FPGAs) to implement algorithms. The main motivation for their development was to achieve both the performance of an 'Application Specific Integrated Circuit' (ASIC), and the flexibility of a microprocessor [15]. As stated in Chapter 1, reconfigurable computing is very promising for applications requiring flexibility as well as real-time response such as robots used in Robo-Cup. These robots require a fast response, and frequent 'personality' changes. , i.e., they require the circuitry to be hard-wired, yet be changeable dynamically.

Pantapolous [62] developed an 'Optically programmable Gate Array' (OPGA). This device integrates electronic and optics to result in more rapid reconfiguration. "The ability to quickly reconfigure allows reconfiguration to be a part of the computation" [62].

Reconfigurable computing hardware has a potential use in RMTs and RMSs. It would allow quick reconfiguration of the control system.

2.4. Survey on 'Setup Planning' methods

A 'Setup' is an instance of the 'machine-workpiece pair'. It gives information on the pose of the workpiece (relative to the machine), fixturing, tooling, manufacturing sequence of features, and machining parameters.

Setup planning is the process wherein decisions are made for:

- Features to be machined in each setup.
- Machining datum(s), fixturing and tooling for each setup.
- Sequence of setups.
- Sequence of operations within setups.

Changing setups incurs cost. Thus, these decisions are made with a goal of minimizing the number of setups, while considering the design and manufacturing constraints.

This thesis focuses on feature-clustering aspect of the 'setup planning' only. Therefore, in this section, research in this area is reviewed.

For feature-clustering, the following criteria/constraints are generally used [16, 27, 52, 61, 69, 93, 110]:

- **Tool approach direction (TAD):** This is defined as the direction in which the tool can approach the feature. Features with same TAD are assigned to the same cluster.
- **Tolerance:** This is defined as the tolerance between the geometric features of the part. Features having tight tolerance are preferably machined in the same setup. This helps ensure better tolerance attributes.
- **Machining precedence:** This is defined as the relations, which dictate the relative machining sequence. Features and setups are sequenced according to the precedence constraints. In addition, cycles may be formed between features or setups. "The presence of cycles yields an infeasible setup plan" [69]. Therefore, for the part to have a feasible process plan, these cycles need to be eliminated. The cycles can be eliminated either by ignoring precedence relations or by splitting feature-clusters.

Ling [45] developed a feature-clustering method. Her method searched for patterns of features, within the part and over the part-family. These feature-patterns were then mapped onto 'gang-spindles-tool head' designs. Her research was an extension of the RMTs to simultaneously machine multiple features.

Ming and Mak [52] developed a method for setup planning. This research proposed to first cluster the features using Kohonen's 'Self-Organizing Map' (SOM). This was followed by sequencing the feature-clusters, and finally, sequencing the machining operations within each feature-cluster. They considered both the sequencing problems as a 'Traveling Salesman Problem' (TSP) and solved them using 'Hopfield neural networks'.

Past research with 'Hopfield neural networks' stated a few general drawbacks of using a Hopfield net to solve the TSP, *viz.*, sub-optimal convergence, convergence on infeasible

solutions, and the variable sensitivity of results to the tuning of the network's parameters[1]. However, Ming and Mak did not report as how their method overcame these limitations.

The neural network would have to be tuned for each part. Their research does not provide any information/ heuristic/ rules of thumb, to determine/ guess the network's parameters. To aggravate this problem, there is no information on how to decide the goodness of the tuning, i.e., the designer cannot judge whether a set of parameters suits the particular part well. Therefore, even 'trial and error' is unlikely to work.

It is also not clear, as how the SOM reacts to cyclic feature-data like $f_1 \rightarrow f_2 \rightarrow f_1$. If not eliminated, such cycles render the process plan infeasible [69]. Their research does not mention about the possibility of cycles between setups. It is unclear whether using the SOM precludes the existence of cycles between setups. If SOM does not preclude cycles, a cycle implies the same setup needs to be visited more than once. This is in direct contradiction with TSP, which is formulated to cover each city only once. Therefore, the correspondence between the sequencing problem and the TSP is imperfect.

There have been reported uses of SOM for solving the TSP [1, 84]. Therefore, a SOM could have feasibly replaced a Hopfield neural network. Using SOM alone to perform the complete method would have been a cheaper proposition. Their motivation to choose Hopfield networks over SOM is unclear.

They represented the feature's manufacturing attributes as vectors (1-D array). These vectors were modeled as a collection of absolute attributes of features and the relative attributes of the different features. These vectors were then clustered using SOM. The SOM algorithm works by computing the distances between the vectors. The part of the vector constituting the 'relative relation' is analogous to distance itself. From the survey [39, 83] no SOM application has been encountered, which takes such a combined input. It is not clear from their research as how they pre-process the data to render the input suitable for the SOM algorithm.

The motivation for using a 6-dimensional vector to represent the TAD is also unclear. The TAD could be represented as a 3-D unit vector. This would reduce the dimensionality of the input space, thus reducing computational costs.

In general SOM applications, the input data is normalized in the period [-1, 1] before implementing the SOM algorithm. This avoids convergence at sub-optimal solutions [39]. Their research did not normalize the input vectors. Considering the input representation scheme, sub-optimal convergence is suspected. Moreover, their research did not provide the parameters of their neural-networks. Therefore, the convergence characteristics of their neural-network cannot be verified.

Wu *et al* [93] performed setup planning in 4 phases: (1) grouping according to tolerance and precision factor; (2) sub-grouping according to TAD (3) assignment of multi-TAD features to one of the groups according to tolerance; and (4) sequencing, splitting/merging clusters according to TADs, tolerance, and precedence. Their research performs "cycle breaking," but the method is naive. The method presented in their research is mainly 'tolerance oriented'.

Sarma and Wright [69] gave a setup planning method for milled parts. The intent of the method developed was to minimize the number of setups and number of tool changes. The precedence relations between features were classified into levels and a graph-theoretic model was used to group features into setups. They used Mc-Cluskey's algorithm [21] to assist assign unique TADs to the features, which have more than one possible TADs.

Their research did not consider the case where the min-cover [21] has more than one implicant. Min-cover is defined as the smallest set of prime implicants [21], which cover the boolean function. If any one of these prime implicants evaluates to *true*, then all the features can be machined using only the directions constituting that implicant. Therefore, there is no need to compute, and store the complete min-cover. Any prime implicant would suffice. Therefore, their research proposes to generate redundant information. It also asserts the necessity of eliminating cycles caused by the existence of precedence relationships between the features. Their method of cycle-elimination was naïve and hence resulted in a greater number of relations being ignored than those with a more intelligent method.

Zhang and Lin [110] used graph theory and matrix theory based methods for setup planning. Their research considered tolerance relations as 'critical' constraints. They

modeled geometric-faces of the manufacturing features as nodes of the graph, and tolerance relations as edges of the graph. Based on TADs and tolerance relations, faces are grouped with an aim to minimize the number of setups. For each setup, a datum surface was selected. In the next step, the setups were sequenced by heuristics. These heuristics maintained the precedence constraints posed by tolerance and natural manufacturing order. Their research performs cycle killing but using a naïve method.

Huang [27] gave a setup planning method for turned components. They identified locating features. Based on these locating features, TAD and tolerances, features were grouped. These setups were then sequenced according to heuristics. Their research did not mention the possible existence of cycles.

Ong, and Nee [61] proposed to use fuzzy logic for determining setups. Their research required the generation of a fuzzy rule-base. The fuzzy inference system would determine the appropriate setups by using fuzzy operations.

Review of research in the field of setup planning revealed that all of the surveyed researches determined the setups for a pre-determined set of machines. On the other hand, the intent of this thesis is to design the machines for the feature-clusters.

2.5. Summary

This chapter reviewed research in the fields of reconfigurability and 'setup planning'. From the review, it has been concluded that there is no existent method to design the RMTs for a part-family. Moreover, in the domain of setup planning, none of the reviewed methods determined the setups with the aim of tailoring a machine for them. This thesis addresses these two problems.

---

[2] Ref: <www.alicebot.org>

3. Design Structure Matrix

3.1. Introduction

For system/product design, qualitative matrix-based methods like the 'house of quality' have been used traditionally. Advancement of matrix-based methods to include quantitative aspects resulted in a new and effective set of matrix-based methods. For example, 'Design of Experiments' generates a 'sensitivity matrix', 'Axiomatic Design' uses a 'design matrix', Design Structure Matrix (DSM), *etc.* Of these advanced matrix-based methods, the DSM has been typically used for modularization and information streamlining [48]. This thesis poses similar requirements. Therefore, the author decided to use DSM.

DSM is mainly used in two domains:

- System analysis and (re)design: - The typical applications include modularization, interface design, 'Failure Mode and Effect Analysis' (FMEA), value-analysis, and reliability-analysis.
- Project-management: - The typical applications include scheduling of activities/resources, team design, cost/time estimation, and information streamlining.

Section 3.2 presents a review of the past research and application of the DSM. In sections 3.3 and 3.4, details about the DSM are presented. As an example presented in Section 3.5, the research documented in this thesis itself is modeled as a design-project and re-structured using the DSM. Finally, in section 3.6, limitations of the DSM are listed, followed by the summary on the chapter in section 3.7.

3.2. Past research and applications

Steward [75] invented the DSM. He developed the DSM representation and the concepts of principal-circuits, shunts, partitioning, and tearing. As an example, he represented a design-process as a DSM, and then manipulated the DSM with algorithms to suggest the restructuring of the process. He also developed a software tool, PSM32 for modeling and manipulating DSMs.

Rogers [66] developed a new tool, 'DEMAID/GA'. This was a major advancement to the DSM operations presented by Steward. This tool uses a two-phased approach. In this first phase, the DSM was partitioned using an expert system developed with CLIPS [59]. The next phase used a genetic algorithm (GA) to minimize the process-cost by re-sequencing the activities within the blocks.

Bradley [3] used the DSM to analyze a few NASA projects *viz.* Pathfinder, NEAR, SAMPEX *etc.* for technological-readiness, and potential failure points and their effects. He tailored the DSM to represent component technology risk factors and called it 'Technology DSM.' This DSM was used to help identify the patterns of system level risk. To represent component dependencies, he again customized the DSM, and called it 'Interface DSM.' This DSM was used to identify the impact of component criticality on the mission operations. Mohan [53] also used the DSM to manage an unmanned flight project.

Dong and Whitney [17] highlighted the unsuitability of DSM for a new design. They proposed to deduce the DSM from the 'Design Matrix' (DM). The DM is constructed by applying the 'Axiomatic Design' (AD) method to a design problem. Their research was a major step towards unifying AD and DSM methods.

Yassine *et al* [106] used DSM to assess rework probabilities. These probabilities were used to simulate the design process. Based on the simulation results, the average cost of production was estimated. In another research, Yassine, and Browning [102] used a DSM for planning simultaneous development of multiple products. Another novel feature of this research was the consideration of resource constraints.

Smith and Eppinger [73] presented the concept that the eigenvalue of the DSM represent the convergence characteristics of the DSM. They concluded that if the spectral radius of the DSM were greater than one, the DSM would not converge. Yassine and Braha [103] reported 'convergence' to be among the four major problems for a design-project, and emphasized that the eigen-characteristics of the DSM reflect the convergence characteristics of the DSM. Based on this fact, the concept of eigenvalues can also be used to evaluate the robustness of the DSM to changes in the external inputs/environment. If the frequency of change to the inputs/environment is greater than

the minimum eigenvalue, the system would never converge. Thus, by analyzing the eigen-characteristics of the DSM, a system can be designed with both system's convergence and its robustness as the design objectives.

Yassine *et al* [105] presented the concept of 'Do it right the first time.' This concept proposed that the designer analyze the partitioned DSM and rewire the process ,i.e., re-engineer the process by adding/ merging/ splitting processes, and/ or redesign the interface/ interaction/ dependencies between the sub-processes to optimize the process design. In a rather weak way, this research binds AD and DSM methods. This is evident especially when an entity is split, because according to the information axiom of AD, the split sub-entities should be minimally coupled. As a shortcoming, this research did not give systematic judgment criteria for judging the potential candidates for re-engineering. Since the eigenvalues of the DSM reflect the convergence properties of the DSM, the eigenvalues can potentially assist the designer to identify the critical entities, which if re-engineered would lead to quicker convergence.

Lockledge and Salustri [46] developed a variant of DSM to represent communication transactions between designers. This is the only application that used a rectangular DSM.

Hilmolla *et al* [26] used the 'Ashby's theory of variation' to re-engineer the sub-systems by adding variations to them. "Ashby's theory states that, a system model or controller can only model or control something to the extent that it has sufficient internal variety to represent" [65].

Their research considered the subsystem to process the current content of the information, while ignored its history. The current author proposes to alleviate this problem partially, by using a system model wherein each subsystem is modeled as a neural network, i.e., the system would be a network of neural-networks (a meta-neural-network). Such a model would be dynamic and learn while in operation. Thus, the flow of information would constantly train the system, i.e., the history of the information flow is implicitly embedded in the network. Adding variation to such a model is equivalent to adding neurons to them. The system would train itself for the variations. Moreover, the asynchronous nature of the model would realistically model 'information-hiding' [104]. This model would also exhibit non-linearity. These characteristics help to identify the

possibility of emergent-chaos [32]. Moreover, such a model would be more realistic, thus leading to better experimentation and ultimately, better systems.

In a very recent research, Yu *et al* [109] used GA for clustering the DSM. The clustering problem was mapped onto an optimization problem. They chose the 'description length' as the objective-function. This objective function was optimized to find the optimal granularity of the system. The 'description length' is analogous to the measure of 'Kolmogorov's complexity' [85]. In other words, their method firstly quantified the Kolmogorov's complexity of the system, and then used a GA to minimize the complexity by re-organizing the system. The results provided by their method were much better than previously developed methods.

It has been observed that none of the surveyed works used DSM to assist in appraising information. DSM helps to identify the criticality of the design activities mainly by analyzing their effect on the convergence characteristics. If correct information about a critical design activity is available beforehand, the process can converge much sooner. The anticipated reduction in cost resulting from availability of this information would help assign a price to that information.

DSM is used to create design teams and communication interfaces [48]. The scope of this application of the DSM can be extended to (re)configure the Management Information System (MIS) for a design-project. In this scheme, an automated software-agent would first convert the information-model provided by the information manager into a DSM. Next, it would analyze the DSM by applying appropriate algorithms. Further, it would determine the optimal information-structure, and finally, (re)configure the MIS to implement this information-structure.

Reportedly, DSM has been used in the domain of software design. Using DSM yielded faster and more efficient software, which lend themselves to easier integration and debugging [78].

3.3. DSM construction and representation

Products, processes, and organizations all lie in the domain of 'complex systems.' The classic approach to (re)designing a complex system is to execute the following steps [5]:

- Decompose the system/process into simpler, more familiar entities/activities.
- Identify the local interfaces/interdependencies/interactions between the entities/activities. These inter-relations between the entities/activities result in the overall system behavior.
- Note the impact of external environment on the system, and the sensitivity of the system to the inputs and outputs.
- (Re)integrate sub-systems using algorithms/human expertise.

Attributes of the DSM:

- A DSM is a matrix representation of a directed graph, i.e., directed graph and DSM are duals of each other. To illustrate this, Figure 3.1 and Table 3-1 present the directed graph and its corresponding DSM respectively.
- Typically (with an exception of the DSM variant used in [46]), a DSM is a square matrix with identical row and column labels.
- The diagonal elements represent the entities/activities.
- Off-diagonal marks represent the dependency/interaction of one entity on another. $Cell_{ij}$ of the DSM holds information on the dependency of $entity_i$ on $entity_j$.
- The cells in DSM can hold boolean values, i.e., '0' = "having a relation" and 'null' (empty cell) = "no relation" or crisp numbers. These crisp numbers are metrics deduced from the type and strength of the dependencies/ interactions, evaluated according to certain scheme. For the DSMs holding crisp numbers, the scheme consolidates all the different interactions between the two entities into a single number. Usually the strongest dependency/ interaction is represented by '0', and weaker ones by higher numbers (the highest one being nine - a typical Saaty's 9-point scale [70]).

- There are three basic building blocks for describing the relationship among system elements: parallel, sequential, and coupled. The coupled subsystems result in "information cycles", i.e., design iterations. The main reasons for iterative information cycles are: (i) changes in information input and (ii) update of shared information.

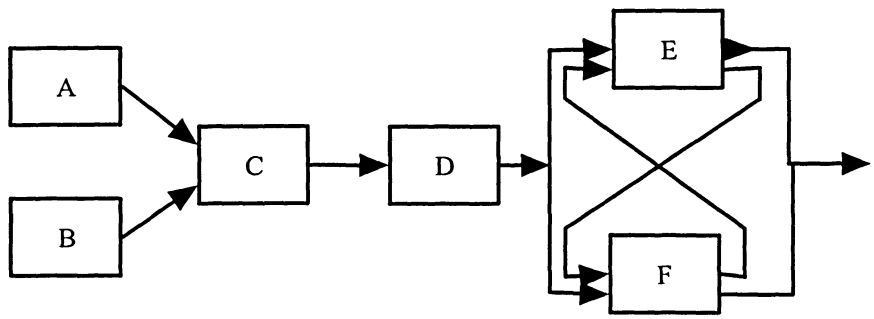


Figure 3.1 Network of activities shown as a graph [3]

Table 3-1 DSM corresponding to the graph presented in Figure 3.1[3]

	1	2	3	4	5	6
1! A	*					
2! B		*				
3! C	0	0	*			
4! D			0	*		
5! E				0	*	0
6! F				0	0	*

3.4. Types of DSMs

There are mainly two types of DSMs *viz.* 'Time based DSM' and 'Static DSM'. These are briefly explained in the following sections.

3.4.1. Time-based (Dynamic) DSMs : In this type of DSM, the ordering of the rows and columns indicates the flow in time. "Upstream activities in a process precede downstream activities" [5]. These DSMs are operated upon by sequencing algorithms. This category of DSM is further divided into 'Activity based DSMs' and 'Parameter based DMSs'. These sub-types of DSMs are presented in sections 3.4.1.1 and 3.4.1.2 respectively.

Definitions pertinent to Time-Based DSMs

Feed-back (-forward) mark: The mark/number above (below) the diagonal corresponding to information sent to a preceding (succeeding) task.

Block: "A Block is the largest set of tasks/activities in which there is a path from every entity to every other in that block and then back again" [75].

Circuit: A circuit is a closed path of activities.

Principal circuit: "A Principal Circuit is the largest closed circuit of activities/tasks in a partitioned block" [75].

Shunt (in context to Block): A shunt is a chain of activities running between entities such that, no link of this chain is coincident with an edge of the Principal circuit.

Operations pertinent to Time-based DSMs

Partitioning: "It is the process of finding 'Blocks' and ordering them such that the predecessor of the block appears before that block" [75].

Tearing: Tearing is the process of choosing certain dependencies, which need to be ignored or assumed, in order to allow the process to proceed. These ignored/assumed dependencies are referred to as tears. Tears are represented either by removing the mark/number corresponding to the assumed dependency or by assigning the dependency a number proportional to the degree of doubt about that assumption. The goal of tearing is to proceed with the process forward by ignoring/assuming a minimum number of dependencies, as late as possible, i.e., the assumption be made just in time, and based on all the previous, although insubstantially information. Generally, the DSM is torn and repartitioned iteratively until the DSM is lower triangular. The assumptions have to be made to allow the process to proceed.

The tearing process involves the following steps:

- Seek tearing advice from designer/heuristics. For details on the heuristics, Ref: [75].
- Consider the advice, and make tears based on engineering knowledge. Tearing is manifested by deleting the dependency mark, or assigning it a number proportional to the degree of doubt about the assumption.

- Re-partition. When partitioning, firstly partition the DSM treating it as a binary DSM, then repartition the blocks formed by temporarily removing the highest numbered marks so on. After complete re-partitioning, repeat the 'tearing process'. Iterative tearing and repartitioning is done until the DSM is rendered lower-triangular. "This usually results in smaller blocks within larger blocks" [75].

Aggregation: It is the process of merging two or more entities into a single entity to eliminate/reduce feedback marks and simplify the DSM. This makes the model less detailed, i.e., the problem is looked upon with a higher level of abstraction.

Decomposition: Typically, the designer chooses a reasonable degree of granularity for representing the entities of a process. Based on the DSM, the designer might choose to un-wrap the sub-entities bundled in an entity and reveal ways to assist the (re)design of the lower-level activities/their interfaces to eliminate/reduce feedback. Representing the system at a lower degree of abstraction is termed, 'decomposition'.

Subtypes of Time-Based DSMs:

3.4.1.1. Activity-Based (or Schedule) DSM: These DSMs are mainly used to improve process structure.

One constructs the DSM by first decomposing the system to the desired granularity, followed by identifying and quantifying the interactions between the entities. The DSM represents the information structure in a compact way. This enhances understanding of the process, which promotes improvement. The main scope of improvement lies in streamlining information and resources, resulting in reduced costs and better utilization.

Unwrapping of the process into entities and analyzing the local interactions helps the designer to understand the process better. It assists in establishing the link between the local information flow, the resulting characteristics, and performance of the process overall. Thus, re-engineering the local interactions has the maximum potential to optimize the overall process.

3.4.1.2. Parameter-Based (or Low-Level Schedule) DSM: These DSMs are used for modeling low-level relationships between design-decisions/ design-parameters/ systems of equations/ subroutine parameter exchanges, *etc.*

The parameter-based DSM is equivalent to an activity-based DSM representing the system at a lower level of abstraction.

This DSM helps determine the sequence of determining design parameters, the sources of iterations, the best start points for the iterative design processes. The main use of these DSMs lies in developing the evolution scheme for detailed design of subsystems or their parameters.

3.4.2 Static DSM

These DSMs represent system elements existing simultaneously, such as components of product architecture or groups in an organization. They are typically operated upon by clustering algorithms. This category of DSM is further divided into 'Component based DMSs,' and 'Team based DSMs'. These sub-types of DSMs are presented in sections 3.4.2.1 and 3.4.2.2 respectively.

Operations pertinent to Static DSMs:

Clustering: It is the processes of clustering entities such that, the entities belonging to the same cluster have maximum interaction between themselves and minimum interaction with elements foreign to its cluster. The DSM clustering is different from the traditional clustering algorithms in only one aspect: the DSM does not require the number of clusters to be input to the clustering algorithm.

Subtypes of Static DSMs:

3.4.2.1. Component-Based (or Architecture) DSM: These DSMs are used for modeling system architectures based on subsystems and their relationships. The local interactions between the interfacing components result in a complex emergent system behavior. Thus, the DSM helps the designer understand the local interactions, and their effects on the system overall. "The DSM is equivalent to the system's genome" [6]. Therefore, re-engineering the system using the DSM is extremely effective, as is its biological counterpart, genetic engineering.

Moreover, a component-based DSM helps approach the problem of modularizing these complex systems by clustering their components systematically. This DSM facilitates

modularization of the system. Modularization reduces the complexity of the system, and the interfaces, thus promoting innovation.

3.4.2.2. Team-Based (or Organization) DSM: These DSMs are used for modeling organization structure based on people and/or groups and their interactions/ communications/ dependencies. Better understanding of organizations enables innovation and improvement in organization design by (re)designing teams and their interfaces.

The DSM is analyzed with the primary intent of restructuring the organization structure. The restructuring is performed by clustering the maximally interacting entities.

3.5. Detailed Example

The research documented in this thesis is design research. The DSM technique is demonstrated by structuring this work. This DSM example assisted to determine the sequence of design activities and the assumptions to be made for the design process to proceed.

Since the construction of a DSM requires detailed knowledge of the process, the DSM shown in Table 3-2 was constructed in retrospect. The author had no prior knowledge about the tasks and information flow before performing the research. Even though the DSM was not useful for structuring the research itself, it is a suitable example to present both a demonstration of the DSM and an early insight into the research.

Table 3-2: The original DSM: representing the flow of information

	1	2	3	4	5	6	7	8	9	10	11	12
1! Part Model	*											
2! Features	0	*										
3! Operation Type		0	*			0						0
4! Optimal TAD		0		*		0						0
5! Precedence			0	0	*							
6! Feature Clustering			0	0	0	*						
7! FR generation						0	*					
8! FR Normalization							0	*				
9! FR clusters							0	0	*			
10! Configuration Generation						0			0	*		
11! Module Design						0				0	*	
12! RMT											0	*

***NB*:** All the DSMs in this thesis are made and manipulated using Steward's software tool[3] (PSM32).

Considering tearing advice and engineering fundamentals, feedback marks are torn iteratively to result in DSM.2 shown in Table 3-3. The tears correspond to the assumptions made. The torn relations are represented by a number corresponding to the degree of doubt about the underlying assumption, or the degree to which the relation can be ignored. These numbers are chosen by the author based on his perspective of the design task.

Table 3-3: DSM 2. The Torn DSM

	1	2	3	4	5	6	7	8	9	10	11	12
1! Part Model	*											
2! Features	0	*										
3! Operation Type		0	*			2						4
4! Optimal TAD		0		*		2						4
5! Precedence			0	0	*							
6! Feature Clustering			0	0	0	*						
7! FR generation						0	*					
8! FR Normalization							0	*				
9! FR clusters							0	0	*			
10! Configuration Generation						0			0	*		
11! Module Design						0				0	*	
12! RMT											0	*

The DSM was used to identify iterations. It also suggested an optimal sequence of design tasks. Most importantly, it suggested making the following assumptions (corresponding to tears):

- The designed RMT is optimal for the chosen set of TADs and operation types.
- The feature-clustering is optimal for the chosen set of TADs and operation types.

The result from the DSM can be extrapolated to: (a) make design teams and (b) export the torn DSM to Microsoft-Project® and generate a detailed schedule, while considering resource constraints *etc.*

---

[3] PSM32-Version 3.1j - Ref: <www.Problematics.org>

A digraph equivalent of the original DSM and the torn DSM are presented in Figure 3.2. The torn relations are shown as dashed lines.

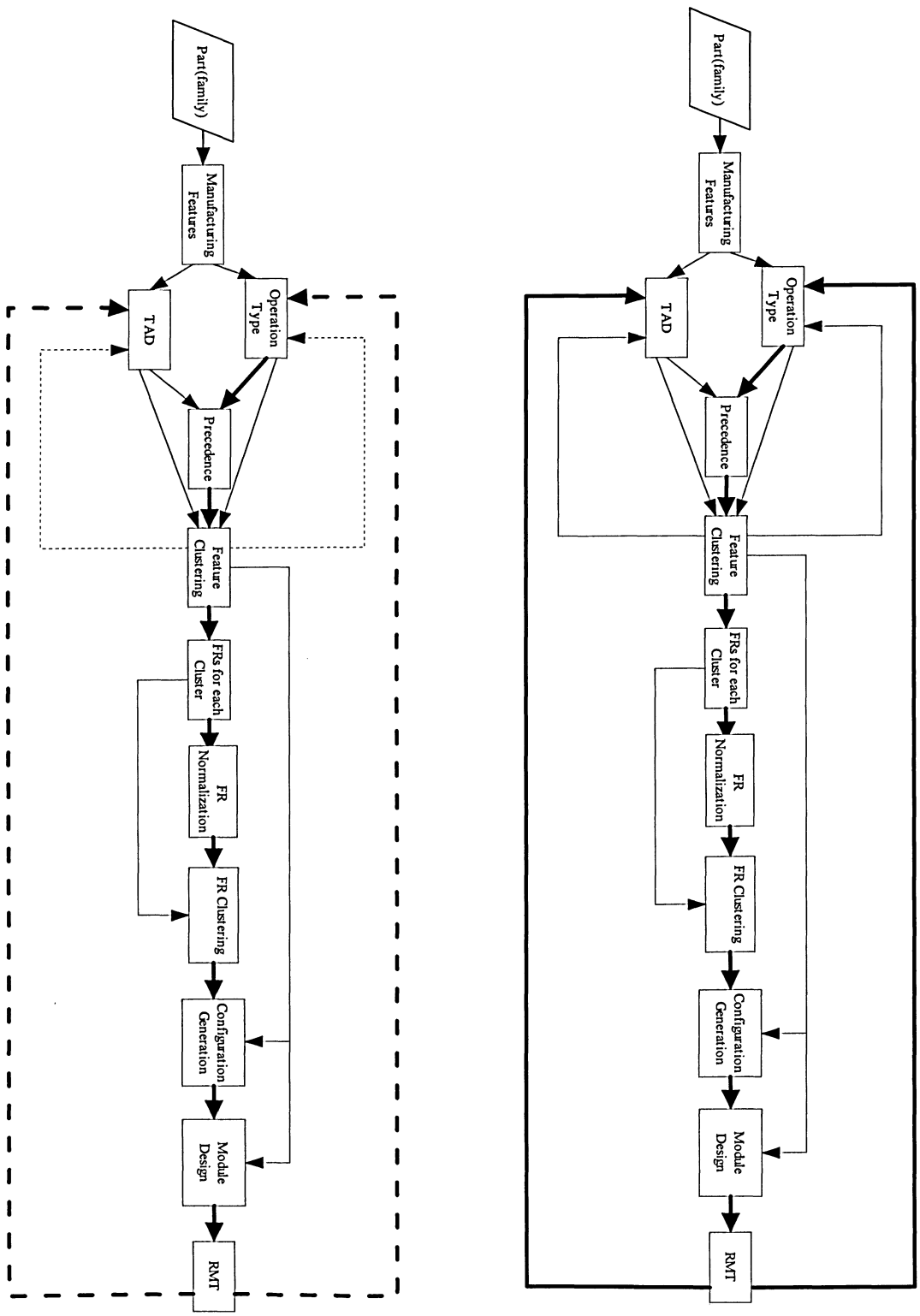


Figure 3.2: (Right) Dual of the original DSM; (Left) Dual of the torn DSM

3. 6. Limitations of DSM

The DSM poses the following limitations.

- A DSM is two-dimensional. It can handle only one metric in a cell. This metric represents a consolidated measure of interdependencies between two entities, instead of representing individual interactions and their corresponding strengths. To represent such a scenario, a 3-D DSM is needed. Higher dimension DSMs are quite complex and have not been developed/investigated in past research efforts.
- A DSM does not show processes overlapping in time [103]. However, a completely torn DSM can be converted into a "Gantt chart," which has the capability to represent overlapping activities. It is mandatory for the DSM to be completely torn before representing the information as a Gantt chart, because a Gantt chart cannot represent iterations.

3.7. Summary

DSM is a strong and versatile matrix tool. It is typically used in system engineering and project management. "DSM is analogous to the genome of the system. [6]" Thus, designing/re-engineering the system by analyzing the DSM is an effective strategy. This chapter surveyed past research in this area and presented a brief tutorial on the DSM.

4. The Method for designing RMTs for machining a part-family

4.1 Introduction

In this chapter, a method to determine the specifications of a set of RMTs for a part-family is developed. As per the approach stated in section 1.4., first a method is developed to determine the specifications of modules and RMTs for manufacturing a single part. Having developed the method for this scaled-down problem, the method is extrapolated to suit the original problem of determining the specifications of a set of RMTs for a part-family. This approach allowed the author to focus on the method itself and not on the complexities arising from 'real-world' problems involving part families.

Section 4.2 presents the development of the method for a single part. This is followed by the extension of this method to address the original problem of determining the functional specification of the RMTs required to machine a part-family. In section 4.3, this extended method is presented.

4.2 The Method for designing RMTs for machining a single part

4.2.1. Introduction

This section develops a method for determining the functional specifications of the RMTs required to machine a single part. The approach for accomplishing this task is presented in section 4.2.2. In the subsequent sections, this approach is followed to develop the method. The resulting method is demonstrated by applying it to an example part.

4.2.2. Approach

Recall that the problem addressed in this section is to determine the specifications of the RMTs required to machine a single part. As stated in section 1.2, this problem decomposes into following sub-problems:

- Determining the specifications for the set of modules, which assemble to form RMT configurations.

- Determining the assembly instructions required to assemble the modules to form the RMTs.

The author approaches the first sub-problem by clustering the manufacturing features of the part, and determining the specifications of the RMT configuration required to machine this feature-cluster. This approach implied that the machining requirements posed by each of these feature-clusters correspond to the functional specification of the respective RMT configurations.

Recall that a RMT has a modular structure. The modules cooperate to fulfill the designated tasks. Therefore, the function requirements (FRs) of the RMTs are equivalent to the 'emergent-behavior' [64] of the assembly of modules. In general, the modules are functionally coupled; therefore, it is very difficult to determine the functional specifications for each individual module. To simplify, it has been assumed that the modules are functionally independent. Having made this simplifying assumption, the process for determining the functional specifications of the modules is developed. This process yields a set of modules. The RMTs required to machine each of the feature-clusters is formed by drawing modules from this 'set of modules' and assembling the modules, into suitable machine configurations.

Having determined the functional definitions for the modules, the interfacing requirements are then identified. These interfacing requirements specify the adjacency of modules.

Based on the functional specification of the modules and their interfacing requirements, they can be assigned physical detail. Although the physical design is beyond the scope of this thesis, yet a detailed proposal on developing an elegant design scheme is presented.

The approach is diagrammatically presented in Figure-4.1.

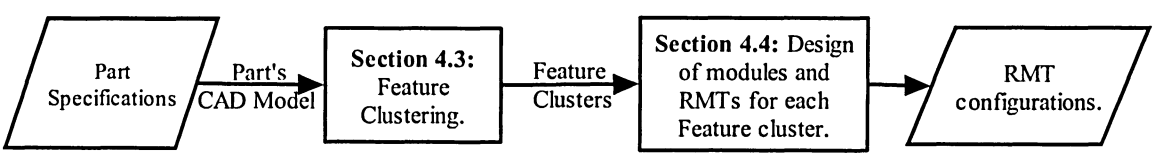


Figure 4.1: The approach for determining the RMT configurations for a single part

4.2.3. The method for 'Feature-clustering'

4.2.3.1. Introduction

From the literature-review, it has been observed that all of the pre-established feature-clustering methods assumed prior knowledge of machining resources. However, in this thesis, the objective is to design a RMT configuration for each of the feature-clusters. The unavailability of a suitable feature-clustering method motivated the author to develop a novel method for feature-clustering.

This section presents a novel feature-clustering method. Brief discussions have also been presented after each step. Having presented the method, it is demonstrated by applying it to an example part.

4.2.3.2. Approach

The part's geometric specifications are converted into manufacturing features.

Similar to the research by Wu *et al* [93], and Sarma and Wright [69], this thesis adopted a two-phased approach for feature-clustering. In the first phase, the features were clustered based on their tool approach directions (TADs), tolerance relations, and machining precedence relations. The precedence relations between the features might result in the existence of a cycle which runs between these initial set of feature-clusters. "Cycles render the process plan infeasible" [69]. Therefore, in the second phase, these cycles are eliminated to result in the final set of feature-clusters.

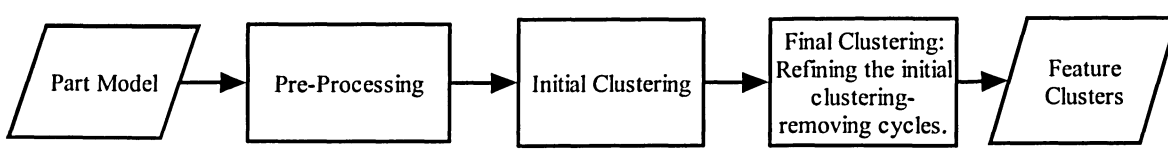


Figure 4.2: The Approach for determining the feature-clusters

4.2.3.3. Pre-Processing

In this pre-processing step, the part's geometric model is converted into a feature-based representation. This task is typically manifested by a 'feature recognition software-agent' like FeatureWorks®[4].

However, this pre-processing step incurs a significant cost. In case the part is designed using feature-based CAD software (like SolidWorks, Catia *etc.*), pre-processing would not be required. Thus, the method would take the input in the same format as the designer designs it. This renders the method more feasible.

In an attempt to simplify, it has been assumed that the part's feature-model is available. This preprocessing step lists the features constituting the part as a set of manufacturing features.

$$F = \{f_1, f_2, f_3, \ldots, f_n\}$$

Where f_i is a manufacturing feature.

The feature representation is intuitive, and encapsulates geometry and machining information. It also has the advantage of being STEP compliant. Most importantly, this representation has the benefit of being free from the part's context.

4.2.3.4. Step1: Identify the TADs for each feature

$\text{Step1}_{\text{Introduction}}$: This step requires human assistance to identify the possible TADs for each feature. As reported in Chapter 2, TAD is one of the basic machining criteria used for clustering features. TAD determines the fixtured pose of the part. For a particular RMT configuration to be able to machine all the features of the cluster, the TAD has to be the same for all the features belonging to the cluster. Therefore, this machining attribute is chosen to be a criterion for clustering features.

$\text{Step1}_{\text{Input}}$: "$F$": The part's manufacturing features.

---

[4] Trademark of Geometric Software Solutions Co. Limited.

Step1$_{Process}$:

- $\forall f \in F$, identify all the possible TADs.
- Generate a new variable f_{tad} and assign it the set: $\{D_1, D_2..., D_p\}$, where D_i is a TAD for the feature, 'f'.
- Augment each feature object with this new variable f_{tad}, i.e., the TAD information is augmented to the information model of the "feature object."

Step1$_{Output}$: "F": the updated set of the feature objects with the information about their corresponding TAD appended to it.

Step1$_{Discussion}$: TADs are not represented as unit vectors as in [134]. Instead, each TAD is considered separately and is given a different name (identifier). The motivation for assigning a unique name to each TAD will explained in the next step.

4.2.3.5. Step2: Determine the optimal TADs for each feature

Step2$_{Introduction}$: Sarma and Wright [69] wrote, "A first step in generating the process plan is the selection of an access direction for each feature. Clearly, the choice of TAD affects the setup in which the feature will be machined, and an important consideration in assigning these TADs is the minimization of number of setups. Unfortunately, however, minimization cannot be performed until after precedence constraints have been generated, and the precedence constraints cannot be generated until TADs have been determined for each feature. Thus, there is a need for iteration."

In this research, the method used for assigning optimal TADs was inspired by Sarma and Wright research [69]. The TADs determined in this step yield a near-minimal solution without iterating. Similar to their approach, this step uses Boolean algebra to determine the minimum set of directions that allow all the features to be machined.

Step2$_{Input}$: Step1$_{Output}$ ("F").

Step2$_{Process1}$: Establish a boolean function for each feature.

For each feature, a 'boolean function' is constructed such that it is *True* (in context to Boolean Logic) for any feasible TAD. This boolean-function has no direct implications and serves only as an input to the following step.

$\forall$ f $\in$ F, establish a Boolean function "f_{bool}" as the 'Boolean OR' of all the directions (D_j) representing the TADs of that feature, i.e.,

$$f_{bool} = \bigvee_{1}^{size(f_{tad})} f_{tad_i} \qquad \rightarrow 4.1$$

Where $\vee$ means 'Boolean OR'.

Physically, "f_{bool}" corresponds to the fact that feature 'f' can be approached in any of its possible TADs. For example, $f_{1_{bool}} = D_1 \vee D_2$ is interpreted as: f_1 can be approached in direction D_1 or D_2.

Step2$_{Process2}$: Construct the boolean equation to represent the machining of the complete part.

The previous process gives the conditions for 'approaching' (in context to machining) each feature. In this step, these boolean functions are combined into an equation to represent the condition for 'approaching' all the features. A solution to this equation would be a set of directions that allow 'approach' to all the features. The minimum sized solution is the smallest number of approach directions that allow the whole part to be machined. The goal of this process is to find a minimal set of directions ensuring the machining of the complete part.

Construct the following equation to represent approach to all the features.

$$\bigwedge_{1}^{n} \left(f_{i_{bool}}\right) = 1 \qquad \rightarrow 4.2$$

Where $\wedge$ is defined as 'Boolean AND'.

Physically, '4.2' represents the condition that all the features be approached.

Step2$_{Process3}$: Find the set of optimal TADs.

The goal of this step is to find the TADs for the features, such that the total number of feature-clusters is minimized. In principle, feature-clusters cannot be formed until the precedence constraints have been identified. However, precedence constraints cannot be identified until the approach direction has been identified. Therefore, to determine the optimal feature-clusters, iteration is required. To avoid this caveat, the optimal TADs are

referred to those TADs, which minimize the number of feature-clusters, while ignoring the dependence of precedence constraints on the TAD. ***NB:*** The optimal TADs do not refer to TADs, which ensure optimality of machining.

This step requires finding the minimum sized solution for Equation 4.2.

As is evident, this is a boolean equation. Thus, boolean algebra techniques are applied. The Left Hand Side (LHS) is in the form of Product of Sums (POS).

The LHS is simplified into a sum of Prime implicants (PI). Prime implicants are a set of product terms involving the minimum number of boolean variable, which if all true result the whole function to be true. The prime implicants of the LHS are generated by using the Espresso software package to simplify the LHS. Espresso is a 2-phase heuristic based boolean minimizer [21] .

Logically, if any of the PIs were 'true', Equation 4.2 would be satisfied, i.e., the complete part can be feasibly machined. For a PI to be 'true', all the literals composing the PI should be 'true', i.e., the part requires approach in all the directions composing the PI in order to machine all the features.

Generally, the PIs are all of the same size, i.e., each involves the same number of variables. In case the PIs are of different sizes, the PI with the minimum number of literals is chosen as the solution. In the event of ambiguity, choose any one PI arbitrarily.

The set of literals constituting the chosen solution corresponds to the minimum-sized set of TADs that permit the machining of the complete part.

E.g., Say the output from Espresso is: - $PI_1 = D_1 \vee D_2$, and $PI_2 = D_2 \vee D_3$

Therefore, 4.2. $\Rightarrow D_1 \vee D_2 + D_2 \vee D_3 = 1$

This implies that all the features can be approached with (D_1 and D_2) or (D_2 and D_3). Therefore, the minimal set of TADs that cover the part (Tad_min) = $\{D_1, D_2\}$ (PI_1 was chosen arbitrarily).

Against general intuition, it is not possible to consider the approach direction as sets of complementary directions like $D_1 = X$ and $\overline{D_1} = -X$. This is because a PI cannot have both a boolean variable and its compliment. In this research, the co-existence of

complementary directions in the minimal set of TADs is allowed, thus the variables are considered individually.

Step2$_{Process4}$: Find the optimal TAD for each feature.

This step requires the designer to identify the optimal TAD for each feature, based on the *Tad_min* set.

f_{direc} = any one of $\{f_{tad} \in Tad\_min\}$.

The optimal TAD for a feature can be any one of the TADs, which also belongs to the *Tad_min* set.

Step2$_{Output}$: "*F*". Each feature object had been augmented with f_{direc} (information on its optimal TAD) and "F" holds these augmented feature objects.

Step2$_{Discussion}$: The main objective of this complete step was to assign TADs to features with more than one possible TAD, such that the total number of feature clusters is minimized. This objective can be achieved in a comparatively simple way as demonstrated by Wu *et al* [93]. The main reason to use a boolean logic based method in this research is to support the possibility of extending the method to implement *fuzzy-logic*, which is a superset of Boolean logic dealing with the concept of partial truth, i.e., truth values between 'completely true' and 'completely false'. A fuzzy-logic based method would not just consider TADs as feasible or infeasible but give a degree of feasibility (of approach) to each direction. This feasibility measure could include the degree of machining optimality offered by each TAD. Thus, the optimal TADs would ensure good manufacturability as opposed to the mere possibility of manufacturing.

The main differences between this step and the method used by Sarma and Wright [69] are:

- This research realizes that only one PI is needed to determine the minimal set of TADs while Sarma and Wright did not mention the course of action in case, the min-cover has more than one PIs. Even though this step proposes to choose any arbitrary PI, it well realizes the redundancy of using the complete min-cover.
- This research uses Espresso instead of Quine-McCluskey as used by Sarma and Wright. Espresso is chosen because Espresso guarantees to solve for the minimum

number of product terms (minimum number of possible TAD sets). It also minimizes the number of literals (minimum sized TAD sets) by using heuristics. Moreover, unlike Quine-McCluskey algorithm, Espresso does not require computational time and memory to be exponentially proportional to the number of inputs.

4.2.3.6. Step 3: Determine rotation attribute for each feature

$Step3_{Introduction}$: This step requires human assistance to identify the rotation requirement: what needs to be rotated to machine each feature: Tool or Work.

Rotation is one of the criteria chosen for feature-clustering. The RMT configuration can have either the work or the tool rotating, but not both because that would cause redundancy, which is undesirable. This step is equivalent to that used by Moon's doctorate [55] wherein, a template is used to determine each feature's 'functional-structure'.

$Step3_{Input}$: "*F*"

$Step3_{Process}$: Identify the rotation attribute for each feature.

$\forall$ f $\in$ F, the designer identifies whether the tool or the work-piece needs to be rotated in order to machine that feature. Rotation of tool corresponds to milling machining function, and rotation of work corresponds to turning machining function.

f_{rot}= 'T' or 'W' depending on whether 'Tool' or 'Work' rotates.

$Step3_{Output}$: "*F*". The feature class has been augmented with f_{rot} and "F" holds these augmented features.

$Step3_{Discussion}$: All the machining features can be machined by either rotating tool or work. According to STEP-NC (also known as ISO-10303-AP:238) [17], operations requiring the tool to rotate correspond to milling, and those requiring the work to rotate correspond to turning machining functions.

Human intelligence is required to decide about the choice of the rotation attribute of the feature.

4.2.3.7. Step 4: Identify, and quantify the manufacturing precedence

$Step3_{Introduction}$: This step requires human assistance to identify and quantify precedence relations. Precedence is inferred from feature modeling, tolerance requirements, natural operation order, and machining best practices.

Precedence relations might lead to cycles. Sarma and Wright stated, "Essentially, the presence of cycles indicates that the current assignment of precedence relations and setups will not yield a feasible process plan" [69]. Thus, cycles have to be eliminated both at the feature level and at the feature-cluster level.

The step uses DSM to eliminate cycles firstly at the feature level and then at the feature-cluster level.

$Step4_{Inpnt}$: "*F*"

$Step4_{Process1}$: Identify and quantify the machining precedence between the features.

Identify the precedence between features and store the information in matrix ***[P]*** such that if f_i precedes f_j then P_{ji}= 'precedence value'. The precedence value is a representative of the criticality of the precedence relation. The precedence value is determined by the quantification scheme presented in Table 4-1. The table is described in detail later.

$\forall$ $\{f_i,f_j\}$ pair, where $f_i,f_j \in F$, identify and quantify the precedence relation according to the scheme given in Table 4-1.

Table 4-1:The Quantification scheme for Precedence relations

Precedence relation	Value
Critical	0
For Quality	1
For Optimality	2

This research assumes the precedence relations to be categorized as *Critical*, *Quality*, and *Optimality*. In the following paragraphs, the criteria to categorize precedence relations are presented.

- "*Critical*" relations are the precedence relations required:
    - For approach to the succeeding feature e.g., if the succeeding feature was made on the bottom face of the preceding one then there is a critical

precedence between them. This precedence type is dependent on the way features are modeled/recognized.

OR

- To maintain the definition of the feature at all times e.g., a blind hole can be made into a thru hole if material is removed from the other end. The requirement to maintain the feature definition asserts a critical precedence constraint.

OR

- For maintaining the explicitly defined inter-feature tolerance relations it is assumed in this research to machine the Datum feature first (if the TADs of the feature and the datum feature are different; else there is no precedence relation).

- Relations supporting "*Quality*," are the precedence relations required: To maintain the feature's internal tolerances. E.g., if a feature has a tight flatness constraint of one of its faces, the machining strategy is to machine that feature as late as possible, without adding a new feature-cluster.

- Relations supporting "*Optimality*," are the precedence relations required to fulfill 'best-machining practices,' which are also based on minimizing machining time, tool wear *etc.*

The more critical the precedence relation, the higher is the cost of ignoring it. Moreover, it has been assumed that the 'critical' precedence relations cannot be ignored. The objective of this step is to eliminate cycles by ignoring the least number of non-critical relations.

In this thesis, the criteria for categorizing precedence relations and the constraints on the cycle elimination step are chosen as the first reasonable scheme. The designer may decide to use a completely different set of criteria and constraints without affecting this process. However, the efficiency of the method would be sensitive to the change in the quantification scheme.

Step4$_{Process2}$: Remove cycles at the feature level.

A DSM is employed to capture cycles and intelligently remove them by deleting the minimal number of non-critical relations. The DSM does not autonomously perform any actions to remove the cycles. It suggests to the designer what relations might be deleted. The tearing advice presented by the software (PSM32) is based on mathematical heuristics. It has no engineering-analysis capability, and is used as a tool by the designer.

Partition the DSM (*[PJ*) using PSM32. The partitioning operation detects the principal-circuits in the blocks formed by the partitioning algorithm.

These principal-circuits need to be broken. For this task, the DSM software is used. Iteratively tear and re-partition the DSM, without ignoring (tearing) any critical relations.

If the cycles cannot be eliminated, it can be inferred that the part cannot be manufactured. Therefore, either an alternative 'feature modeling' scheme be used or the design be modified to render the part manufacturable. Figure 4.3 and Figure 4.4 show an example given by Sarma and Wright [69] to demonstrate the case of a part rendered non-manufacturable because of a cycle existing at the feature level.

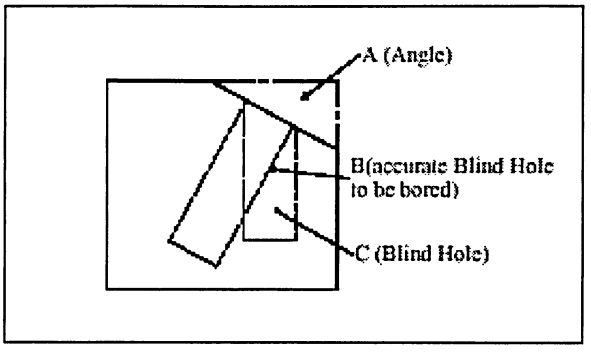


Figure 4.3: An example part, with cycles existing at the feature level [69]

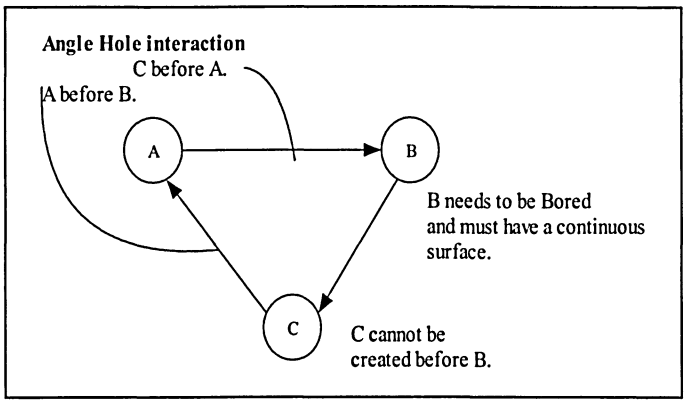


Figure 4.4 : The feature graph corresponding to the part presented in Figure 4.3 [69]

For this example, Sarma and Wright [69] proposed to make a "seating Step" as a change to the part's design. This change avoids 'Angle-Hole interaction', and renders the part manufacturable.

Step4$_{Output}$: This step would output ***[P]***, the precedence matrix, with the relations causing the cycles been deleted, i.e., the precedence relations causing cycles have been ignored. However, in case the cycle could not be eliminated, this step would lead to a conclusion that the part is non-manufacturable. Such a situation would require either choosing a different modeling scheme, or redesigning the part.

4.2.3.8. Step5: Cluster features

This step yields the final set of feature-clusters. Each of these feature-clusters maps to a RMT configuration.

The clustering process is performed in two phases. In the first phase, features are clustered based on their TADs and information on whether the tool rotates or the work. In the next phase, these cycles are removed either by ignoring some precedence relations or by splitting clusters.

Step5$_{Input}$: ***"F"***

Step5$_{Process}$: Cluster Features.

Step5$_{Process1}$: Initial Clustering: group the features with the same TAD and rotation attributes.

To form the initial set of feature-clusters, features with the same TAD and rotation attribute are grouped together to yield the set of initial clusters.

Form the feature-clusters C_i, such that:

- $C_i \subset F$
- $\forall f \in C_i, f_{direc}, f_{rot}$ is identical for all the features in the cluster.
- $C_i \cap C_j = \phi$ (null set) (for $i \neq j$) ,i.e., no overlaps between clusters are allowed.

Here, C_i is a cluster of features, for i=1,2,...

Based on the initial clustering, construct a hyper-graph (cluster-graph) [69]. In this cluster-graph, each of the hyper-nodes corresponds to a feature-cluster C_i, and a hyper-edge (between two hyper-nodes) bundles all the parallel edges, which represent the feature precedence relations between features of two clusters. The hyper-graph is equivalent to a feature-graph, but at a higher degree of abstraction. To demonstrate the concept of a hyper-graph, Figure 4.5 presents a hypothetical graph. This graph represents the initial set of clusters and the precedence relations between the features of the clusters. Figure 4.6 presents the hyper-graph corresponding to the graph presented in Figure 4.5.

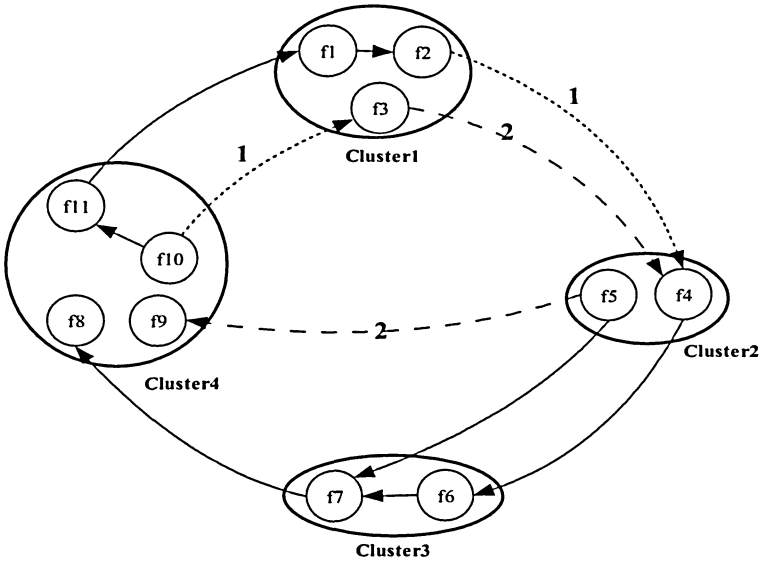


Figure 4.5: A hypothetical feature graph

In Figures 4.5 and 4.6, the solid, dotted, and dashed lines correspond to 'strength-0', 'strength-1', and 'strength-2' relations respectively.

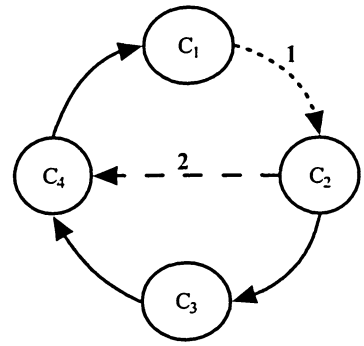


Figure 4.6 : Hyper-graph corresponding to the graph presented in Figure 4.5

The hyper-edges encapsulate all the edges (relations) between the clusters. The strength of the hyper-edge is the strength of the strongest edge it encapsulates , i.e., the one with the lowest metric (because the lower the metric, the higher the criticality).

$Strength(HyperEdge(C_i \rightarrow C_j)) = Min(P[ii, jj])$ where $f_{ii} \in C_i$, $f_{jj} \in C_j$, and

[P] is the precedence matrix.

The cluster-graph represents the precedence-graph with a higher degree of abstraction. A DSM is constructed from the cluster-graph in the next step. This DSM then undergoes the process of intelligent cycle elimination.

Step5$_{Process2}$: Removing cycles from the initial set of clusters

The initial set of clusters is analyzed for cycles. Eliminating these cycles is necessary to yield a feasible manufacturing plan. The cycles can be broken either by deleting a hyper-edge or by splitting clusters. Deleting a hyper-edge implies that all the precedence relations bundled in it are ignored. The DSM is used for intelligently suggesting the edges (which correspond to the precedence relations) to be broken in order to remove cycles. If the edges were broken, (i.e., the precedence relations were ignored) without any intelligent method, more number of edges would have to be broken as compared to those by using an intelligent method.

If a cluster must be split, the resulting clusters should be of similar sizes. Recall that each of the resulting clusters correspond to a RMT configuration. Balancing the split sub-clusters would ensure that the resulting RMTs be balanced in terms of workload. This

would balance the machining system on a whole and would reduce the possibility of creating bottlenecks.

Due to the nature of the problem, the designer cannot get any information to judge the potential benefit achieved by splitting any particular cluster. Even the DSM cannot assist the designer for this task. Therefore, it is proposed to consider splitting clusters in decreasing order of their sizes. Although this does not guarantee similarly sized clusters, it does contribute towards this objective.

The method for removing cycles at the feature-cluster level

Pre-Processing: In this phase, the Cluster-Graph from the previous step is converted into its DSM dual. Recall that the correspondence between a graph and a DSM has been asserted in Chapter 3.

The resulting DSM is partitioned by PSM-32. Based on the heuristic embedded in it, the software assigns a metric to each of the edges of the principal-circuit. These metrics represents the potential benefit of ignoring a relation between two entities. These metrics are assigned solely on the mathematical basis, and do not reflect any form of engineering decisions. In other words, these metrics represent the 'goodness of tears'. The lower the metric, the higher is the probability of removing the cycle by ignoring that relation.

Phase -1: Here, the intent is to eliminate cycles by ignoring non-critical precedence relations. This subroutine looks for potential tears in decreasing order of the 'goodness of tears'. The lower the 'goodness of tearing', the lower is the probability to eliminate the cycle by ignoring that relation. Ignoring a relation is equivalent to deleting it from the precedence matrix, ***[P]***.

Pseudo code:

/*Phase 1- Eliminate cycles by cutting edges*/

'tear_variable':= 'best tear' /* *Hold the cell having the best 'goodness of tear' in the variable: 'tear_variable'* */

While {(cycle ≠ Φ) OR (all cells in the principal-circuit have been covered)}

if (Strength of the 'tear_variable'≠ Critical)

then

break that hyper-edge. /**delete a hyper-edge⇒ ignore all relations bundled in it. ⇒ the corresponding relation in [P] are deleted*/*

else

'tear_variable' = next-best tear.

end

end /**Phase-1 ends*/*

Discussion: As has been mentioned earlier, the heuristics embedded in the software assign a metric to each of the cells, which are a part of the principal-circuit (defined in Chapter 3). The metric assigned to a cell corresponds to the potential of eliminating the principal-circuit by ignoring the relation represented by that particular cell. Therefore, the lower the 'goodness of tear', the lower is the chance for the cycle to be eliminated by ignoring that relation. Ignoring relations with the 'goodness of tear' being below a certain threshold would have a very low probability of eliminating the cycle. This possibility is not considered in this work.

If (cycle ≠ Φ) /**if the cycle exists*/*

then

terminate

else

execute 'Phase-2'

Phase -2: The option of eliminating the cycles by ignoring relations has been exhausted in Phase-1. Therefore, in order to eliminate the cycles, the clusters ought to be split. However, the clusters can be split with or without ignoring any relations between the features. In this phase, the clusters are split without ignoring any relations (breaking any edge).

Pseudo Code:

/*Phase2: Split clusters without cutting any internal edges*/

"cluster_variable" := the largest un-considered cluster in the cycle.

While {("cluster_variable" ≠ Φ) OR (cycle≠ Φ)}

```
        if ({in[5]} ∩ { out[6]} =Φ)
        then
            "cluster variable" := the largest un-considered cluster in the cycle.
        else
            if (({in}∪ {in.child[7]}) ∩ {out} =ϕ)
            then
                1st Sub_cluster= ({in}∪ {in.child})
                2nd Sub_cluster =cluster-1st Sub_cluster
                goto Pre-Processing
            else
                continue
            end
        end
    end /*Phase-2 ends*/
```

Discussion: Recall that each feature-cluster corresponds to a RMT configuration. Therefore, if the number of clusters increases, the RMT would have to be reconfigured more times. This phase increases the number of clusters. Hence, this phase is implemented after the option of ignoring non-critical relations has been exhausted. To balance the workload on the RMTs, the final set of clusters should be balanced in terms of size. To partially fulfill this objective, the clusters (to be split) were considered in the decreasing order of their sizes.

```
If (cycle≠ Φ) /*if the cycle exists*/
then
    terminate
else
    execute 'Phase-3'
```

---

[5] Nodes, which have the edges corresponding to the Hyper-edge of the cycle running into them.
[6] Nodes, which have the edges corresponding to the Hyper-edge of the cycle running out of them.
[7] Nodes having an edge running into them from the nodes belonging to the 'in' set.

Phase -3: In the previous phase, the clusters were split without ignoring any precedence relations. This phase is an extension of the Phase-2, in that the clusters are split by ignoring relations. The author proposed to use a constrained-min-cut algorithm [76] for this task. The constraint being that critical constraints cannot be ignored.

Pseudo Code:

/*Phase3: Split clusters by cutting internal edges*/

While {(Cycle≠ Φ} /*if the cycle exists*/

Pick the largest un-considered cluster (belonging to the cycle),

If ({***in***} ∩ {***out***} =Φ)

then

consider the cluster as a graph.

Partition the graph using constrained min-cut algorithm such that

{***in***}⊂ 1st partition

{***out***}⊂ 2nd partition

No 'Strength-0' edges are cut.

/*done manually single-handedly or with DSM assistance*/

goto Pre-Processing

else

continue

end

end */*Phase-3 ends*/*

If (cycle≠ Φ) /**if the cycle exists*/*

then

terminate

else

throw exception ("part is infeasible for process planning")

end

Discussion: In this phase, a constrained min-cut algorithm is used. This algorithm splits the clusters by ignoring the minimum number of non-critical relations. Moreover, it balances the resulting sub-clusters in terms of size. This algorithm is used as a black-box.

However, if the cycle remains, it can be concluded that the part leads to an infeasible process plan [69].

$Step5_{Output}$:

The set $C=\{C_i\}$ representing the new set of clusters and an updated set of hyper-edges connecting them (an updated ***[P]***.)

$Step5_{Discussion}$:

In this step, the cycles were eliminated intelligently. Using the DSM based scheme eliminates the cycles by ignoring a lesser number of relations. This method results significantly better than those by Sarma, and Wright's method [69]. The comparison between the two is presented in section 4.2.3.10.

4.2.3.9. Application of the method to the example part to yield feature-clusters

1a. Input: -The Part Model with all the tolerances and dimensions specified. D_is are the TADs. All the measurements are in millimeters. Standard geometric-tolerance symbols are used.

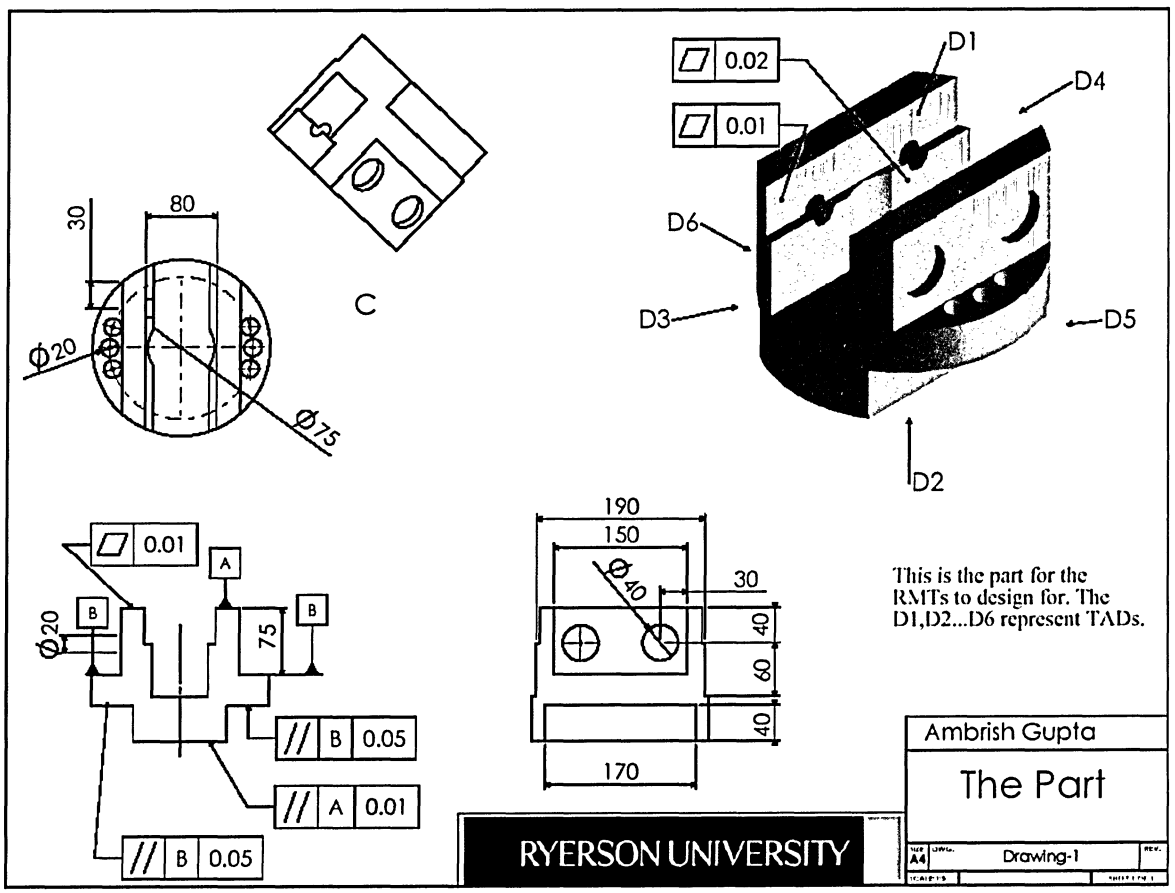


Figure 4.7: The Part

1b. Input:-The Raw Stock. The method implicitly requires the Raw-Stock as the Input. Figure 4.8 presents the drawing of the Raw-Stock, and specifies its dimensions. The raw-stock is assumed to be squared, i.e., the end faces are planed.

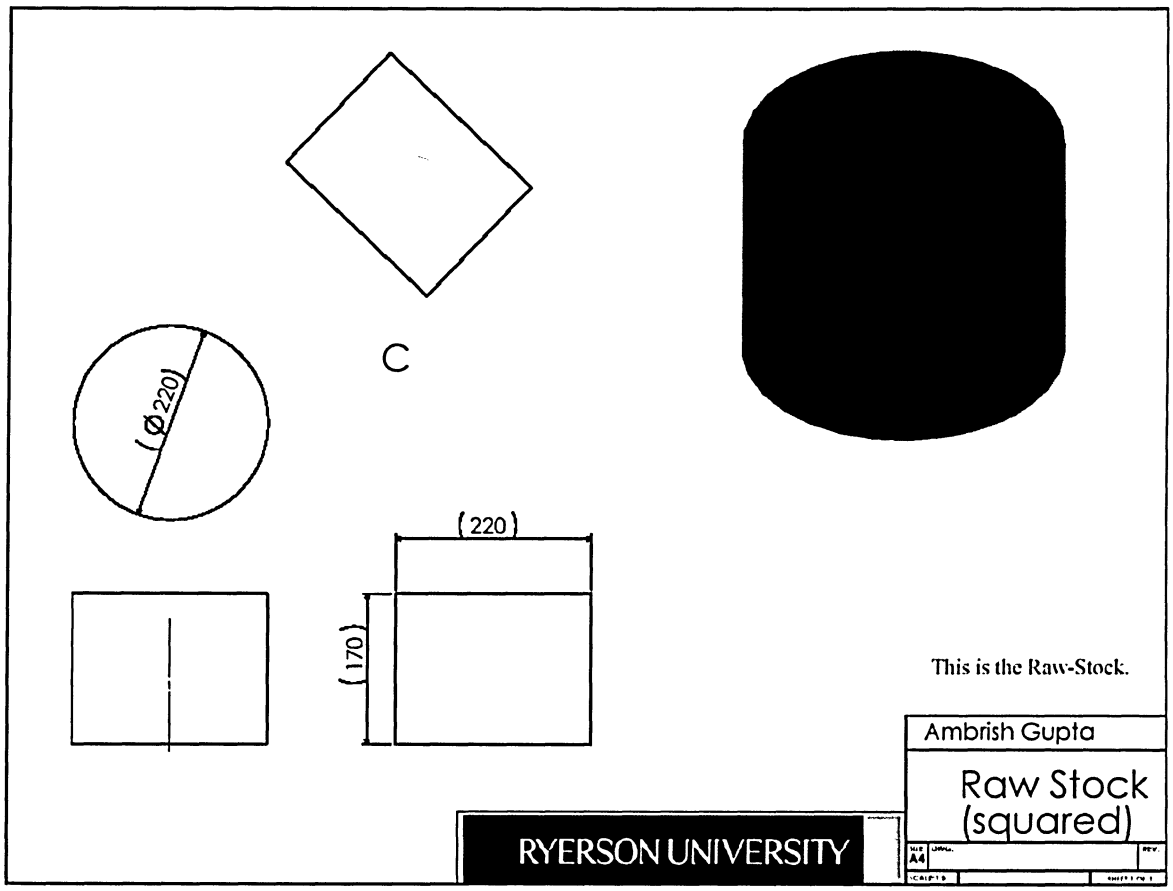


Figure 4.8: The Raw Stock

2a. Preprocessing: - The features are recognized using an external "feature recognition software-agent" like FeatureWorks®. The features are presented in Figure 4.9. The names of the features are compliant with the STEP-AP-224.

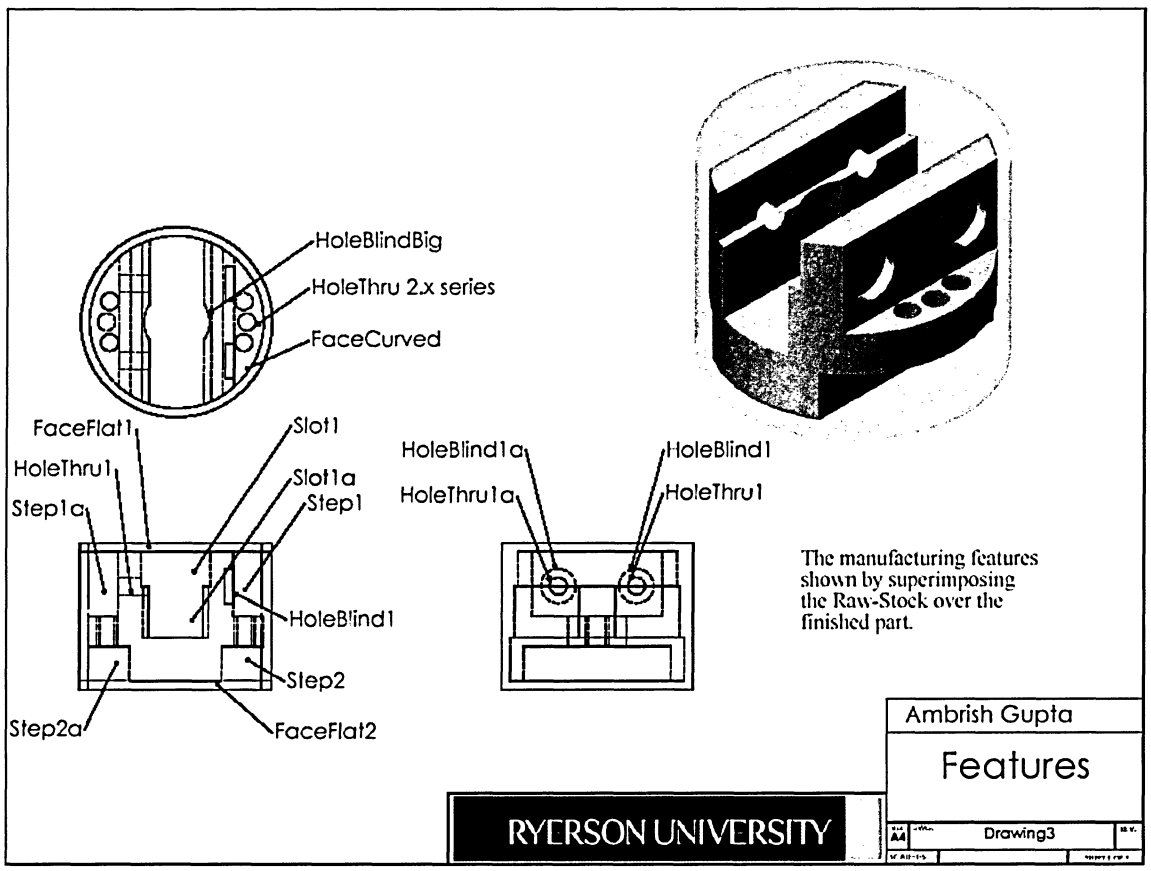


Figure 4.9: Features

2b.3. Preprocessing, identification of Rotation attribute: - The TADs and rotation attribute are identified for each feature. This information is presented in Table 4-2. The table also shows the f_{bool} for each feature. Based on f_{bool} of the features, Equation 4.2 is constructed. This equation is then simplified using Espresso. The simplification of the boolean function on the LHS leads to a single resulting Prime implicant. The literals constituting this prime implicant correspond to the ***Tad-min*** set. The intersection of ***Tad-min***, and f_{bool} for each feature results in f_{direc}, the optimal TAD.

Table 4-2: Features, their corresponding TADs, and Rotation characteristics (Y=YES)

#	Feature	Rotation		TAD						f_{bool}	Eqn-4.2	Eqn-4.2 Simplified using	f_{direc}
		Tool	Work	D1	D2	D3	D4	D5	D6				
1	$Face_{curved}$		Y	Y	Y					(D1∨D2)	(D1∨D3∨D4∨D5)∧(D1∨D3∨D4∨D6)∧(D2∨D3∨D4∨D5)∧(D2∨D3∨D4∨D6))=1	D1∧D2∧D4∧D6=1; Therefore approach in these directions can machine all the features of the part.	D1
2	$Face_{Flat1}$		Y	Y						(D1)			D1
3	$Face_{Flat2}$		Y		Y					(D2)			D2
4	Step 1	Y		Y		Y	Y		Y	(D1∨D3∨D4∨D6)			D1
5	Step 1a	Y		Y		Y	Y	Y		(D1∨D3∨D4∨D5)			D1
6	Step 2	Y			Y	Y	Y		Y	(D2∨D3∨D4∨D6)			D2
7	Step 2a	Y			Y	Y	Y	Y		(D2∨D3∨D4∨D5)			D2
8	Slot 1	Y		Y		Y	Y			(D1∨D3∨D4)			D1
9	Slot 2	Y		Y		Y	Y			(D1∨D3∨D4)			D1
10	$Hole_{blind}$Big	Y		Y		Y	Y			(D1∨D3∨D4)			D1
11	$Hole_{blind}$1	Y						Y		D5			D5
12	$Hole_{blind}$1a	Y						Y		D5			D5
13	$Hole_{thru}$1	Y							Y	D6			D6
14	$Hole_{thru}$1a	Y							Y	D6			D6
15	$Hole_{thru}$2.1	Y		Y						D1			D1
16	$Hole_{thru}$2.2	Y		Y						D1			D1
17	$Hole_{thru}$2.3	Y		Y						D1			D1
18	$Hole_{thru}$2.4	Y		Y						D1			D1
19	$Hole_{thru}$2.5	Y		Y						D1			D1
20	$Hole_{thru}$2.6	Y		Y						D1			D1

3. Precedence Constraints:

Table 4-3 presents the precedence relations between the features. These relations were quantifies according to the scheme presented in 4.2.3.7.

Table 4-3: Feature Precedence

Predecessor	Successor	Relation	Value	Explanation of Relation
Face_flat1	Face_flat2	Critical	0	Parallelism defined explicitly for Face_flat2 wrt. Face_flat1. Therefore, Face_flat1 need to be machined before Face_flat2.
Face_flat2	Step 1, Step 1a, Slot 1, Slot 2	Critical	0	For approach.
Step 1,Step 1a	Slot 1, Slot 2:	Quality	1	For maintaining flatness in the slot features.
Slot 1	Slot2, HoleblindBig	Critical	0	For approach
Step1 ,Step1a	Step2, Step2a	Critical	0	Parallelism defined explicitly for Step 2, Step 2a wrt. Step 1, Step 1a.Therefore, Face_flat2 need to be machined before Step 2, Step 2a.
Slot2	HoleblindBig	Optimal	2	Best machining practice: machine shallow features before deep.
Step 1, Step 2	Holethru2.1,2.2, 2.3	Critical	0	For definition
Step 1a, Step 2a	Holethru2.4,2.5, 2.6:	Critical	0	For definition
Step1	Holethru1,1a	Critical	0	For approach
Slot1,Slot2	Holethru1,1a	Critical	0	For definition
Step 1a	Holeblind1,1a	Critical	0	For approach

4. Identify cycles

Table 4-3 is translated into the DSM format (presented in Table 4-4). This DSM assists the designer to identify cycles existing at the feature level, and further at the feature-cluster level. The numerals in the DSM represent the strength of the precedence relations between features (0 = Critical; 1 = for Quality; 2 = for Optimality).

Table 4-4: DSM 1. The Original DSM for the Features Vs. features

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1! Facecurved	*																			
2! FaceFlat1		*																		
3! FaceFlat2		0	*																	
4! Step 1			0	*																
5! Step 1a			0		*															
6! Slot 1			0	1	1	*														
7! Slot 2			0	1	1	0	*													
8! HoleblindBig						0	2	*												
9! Holethru2.1					0				*							0				
10! Holethru2.2					0					*						0				
11! Holethru2.3					0						*					0				
12! Holethru2.4				0								*			0					
13! Holethru2.5				0									*		0					
14! Holethru2.6				0										*	0					
15! Step 2				0	0										*					
16! Step 2a				0	0											*				
17! Holeblind1					0												*			
18! Holeblind1a					0													*		
19! Holethru1				0		0	0												*	
20! Holethru1a				0		0	0													*

This DSM (Table 4-4) is firstly used to identify cycles at the feature level. The DSM is partitioned using PSM32, resulting in Table 4-5. From this table, it can be observed that no cycles exist at the feature level.

Table 4-5: DSM2. The Partitioned precedence matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1! Facecurved	*																			
2! FaceFlat1		*																		
3! FaceFlat2		0	*																	
4! Step 1			0	*																
5! Step 1a			0		*															
6! Slot 1			0	1	1	*														
17! Holeblind1					0		*													
18! Holeblind1a					0			*												
7! Slot 2			0	1	1	0			*											
15! Step 2				0	0					*										
16! Step 2a				0	0						*									
8! HoleblindBig						0			2			*								
9! Holethru2.1					0						0		*							
10! Holethru2.2					0						0			*						
11! Holethru2.3					0						0				*					
12! Holethru2.4				0						0						*				
13! Holethru2.5				0						0							*			
14! Holethru2.6				0						0								*		
19! Holethru1				0		0			0										*	
20! Holethru1a				0		0			0											*

Having confirmed that there are no cycles at the feature level, the method proceeds to form the initial set of feature-clusters. These clusters are formed by grouping together the features with identical f_{rot} and f_{direc}.

Table 4-6, represents the initial feature-clusters as a colored block along the diagonal. This representation helps construct the DSM corresponding to the hyper-graph, which is used to identify the existence of cycles at the feature-cluster level.

Table 4-6: DSM 3. The initial feature-clusters

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1! Facecurved	*																			
2! FaceFlat1		*																		
3! FaceFlat2		0	*																	
4! Step 1			0	*																
5! Step 1a			0		*															
6! Slot 1			0	1	1	*														
7! Slot 2			0	1	1	0	*													
8! HoleblindBig						0	2	*												
9! Holethru2.1					0				*							0				
10! Holethru2.2					0					*						0				
11! Holethru2.3					0						*					0				
12! Holethru2.4				0								*			0					
13! Holethru2.5				0									*		0					
14! Holethru2.6				0										*	0					
15! Step 2				0	0										*					
16! Step 2a				0	0											*				
17! Holeblind1					0												*			
18! Holeblind1a					0													*		
19! Holethru1				0		0	0												*	
20! Holethru1a				0		0	0													*

Based on Table 4-6, Table 4-7 is constructed. Each cell of Table 4-7 corresponds to a cluster formed after the initial clustering stage. Partitioning the DSM (Table 4-7), highlights the cycle: $Cluster_3 \rightarrow Cluster_4 \rightarrow Cluster_3$. To render this part manufacturable, this cycle needs to be removed.

Table 4-7:DSM4. Clusters Vs Clusters (shows a cycle with critical relations)

	1	2	3	4	5	6
1! Cluster1	*					
2! Cluster2	0	*				
3! Cluster3		0	*	0		
4! Cluster4		0	0	*		
5! Cluster5			0		*	
6! Cluster6			0			*

It can be observed from Table 4-7 that the cycle comprises of 'Strength-0' relations (Critical relations). Since ignoring these relations are not allowed, the feature clusters need to be split. Following the method (to remove cycles) presented in 4.2.3.8., the cycle is removed, and the resulting feature-clusters are presented in Table 4-8.

Each feature-cluster is presented on the diagonal. C_1 = [*Face curved*, *FaceFlat1*], C_2 = [*FaceFlat2*], C_3 = [*Step 1*, *Step 1a*, *Slot 1*, *Slot 2*, *HoleblindBig*], C_4 = [*Step2*, *Step2a*], C_5 = [*Holeblind1*, *Holeblind1a*], C_6 = [*Holethru1*, *Holethru1a*], C_7 = [*Holethru2.x*] $_{(x=1,\ldots 6)}$.

In Table 4-8, the feature-clusters are presented on the diagonal, and enclosed with different boundary types.

Table 4-8: DSM 5. The final feature-clusters

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1! Face curved	*																			
2! FaceFlat1		*																		
3! FaceFlat2		0	*																	
4! Step 1			0	*																
5! Step 1a			0		*															
6! Slot 1			0	1	1	*														
7! Slot 2			0	1	1	0	*													
8! HoleblindBig						0	2	*												
15! Step 2				0	0				*											
16! Step 2a				0	0					*										
17! Holeblind1					0						*									
18! Holeblind1a					0							*								
19! Holethru1				0		0	0						*							
20! Holethru1a				0		0	0							*						
9! Holethru2.1					0					0					*					
13! Holethru2.2				0					0							*				
14! Holethru2.3				0					0								*			
10! Holethru2.4					0					0								*		
12! Holethru2.5				0					0										*	
11! Holethru2.6					0					0										*

Feature Cluster Drawings

Figure 4.10- 4.16 present the part after the machining of the feature-clusters.

Figure 4.10 presents the part after Cluster-1 (C_1 = [*Face curved, FaceFlat1*]) has been machined.

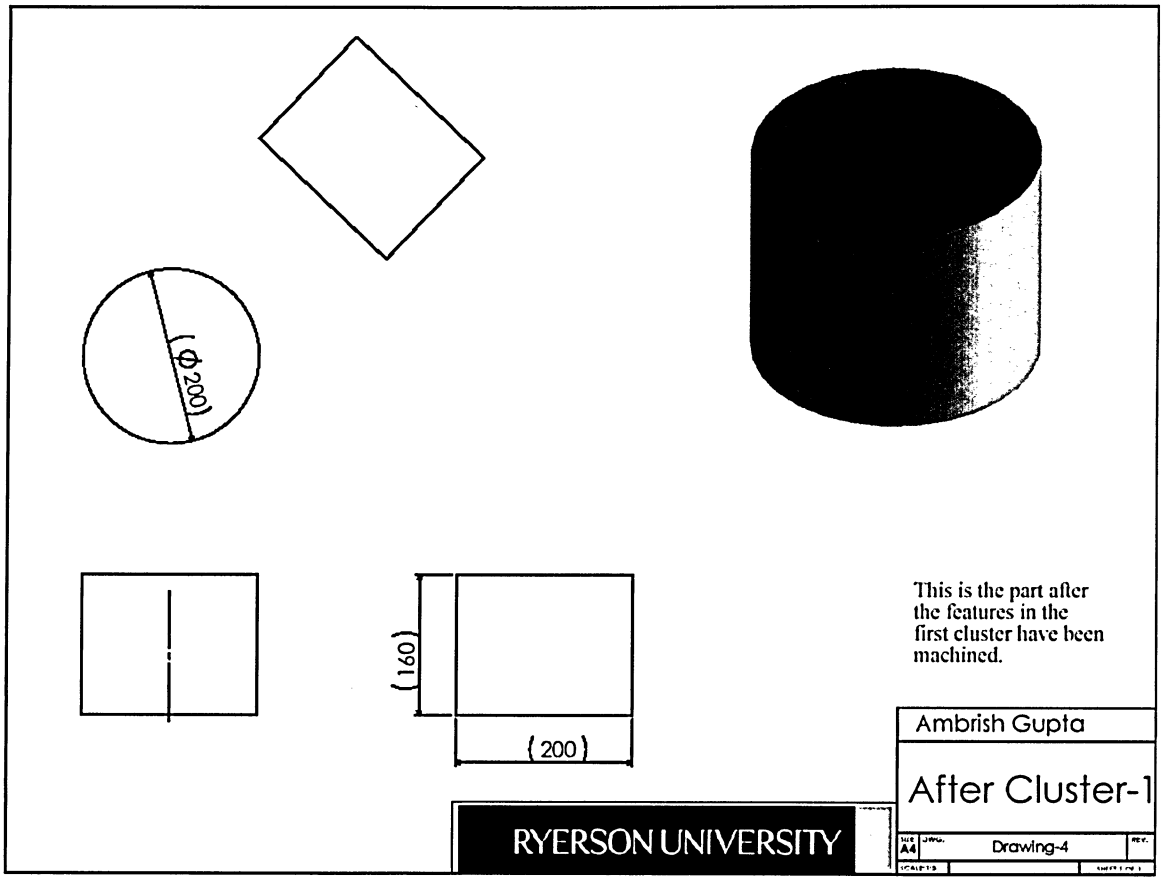


Figure 4.10: After machining feature-cluster-1

Figure 4.11 presents the part after Cluster-2 (C_2 = [*FaceFlat2*]). Machining *FaceFlat2*, requires shaving off 10mm from the flat face.

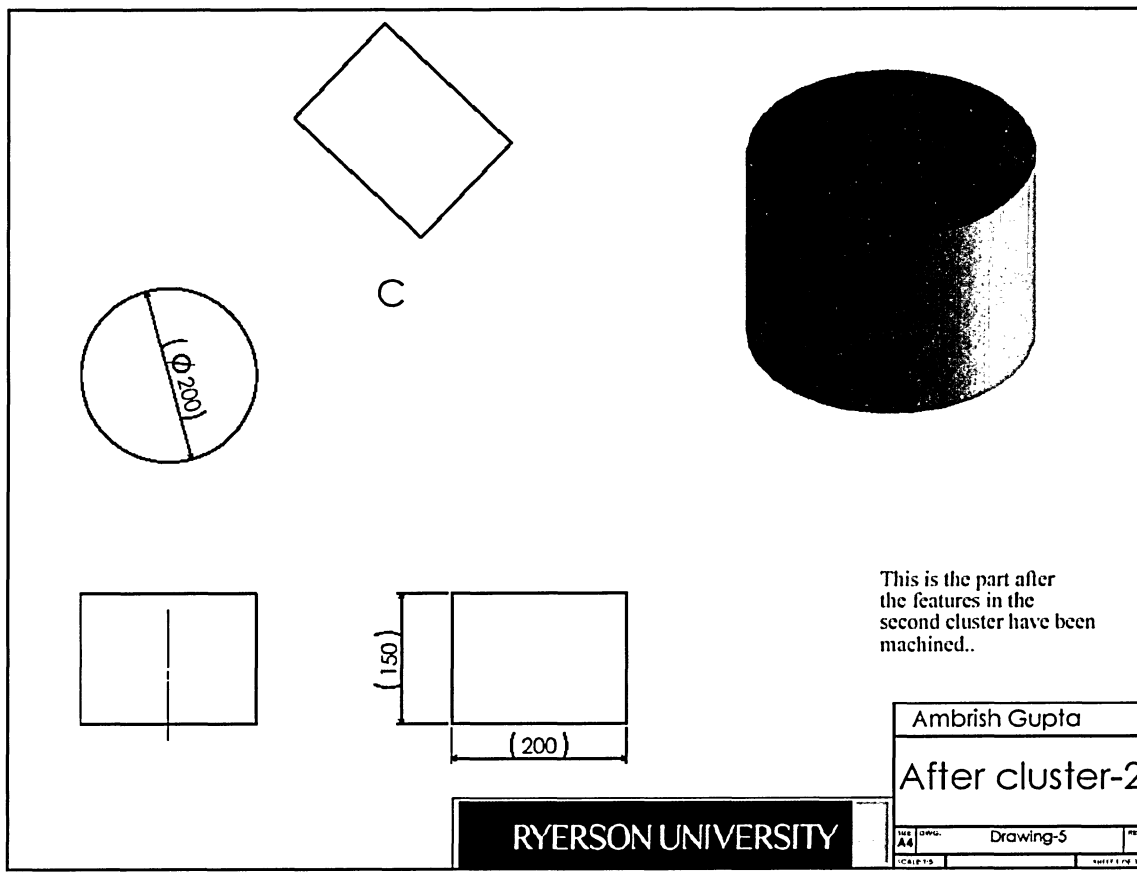


Figure 4.11: After machining feature-cluster-2

Figure 4.12 presents the part after Cluster-3 (C_3 = [*Step 1*, *Step 1a*, *Slot 1*, *Slot 2*, *HoleblindBig*]) has been machined.

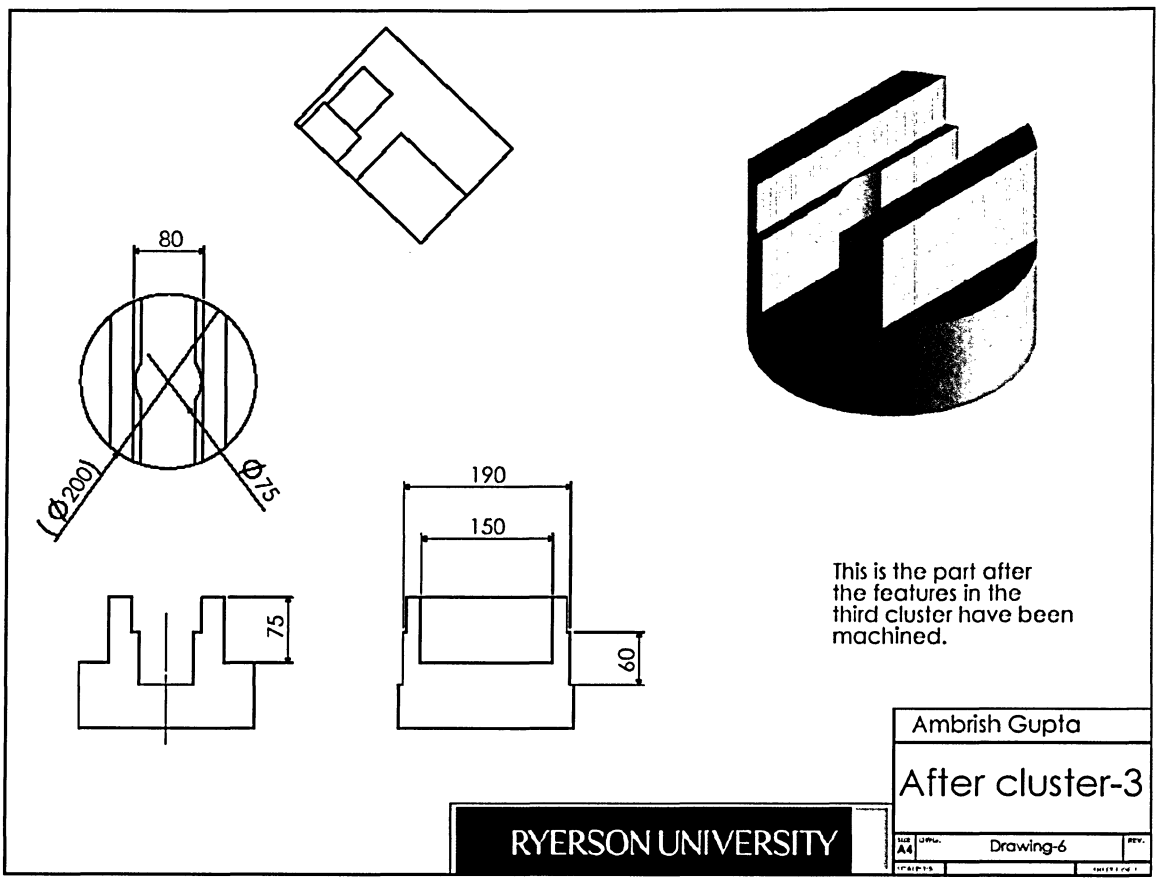


Figure 4.12: After machining feature-cluster-3

Figure 4.13 presents the part after Cluster-4 (C_4 = [*Step2*, *Step2a*]) has been machined.

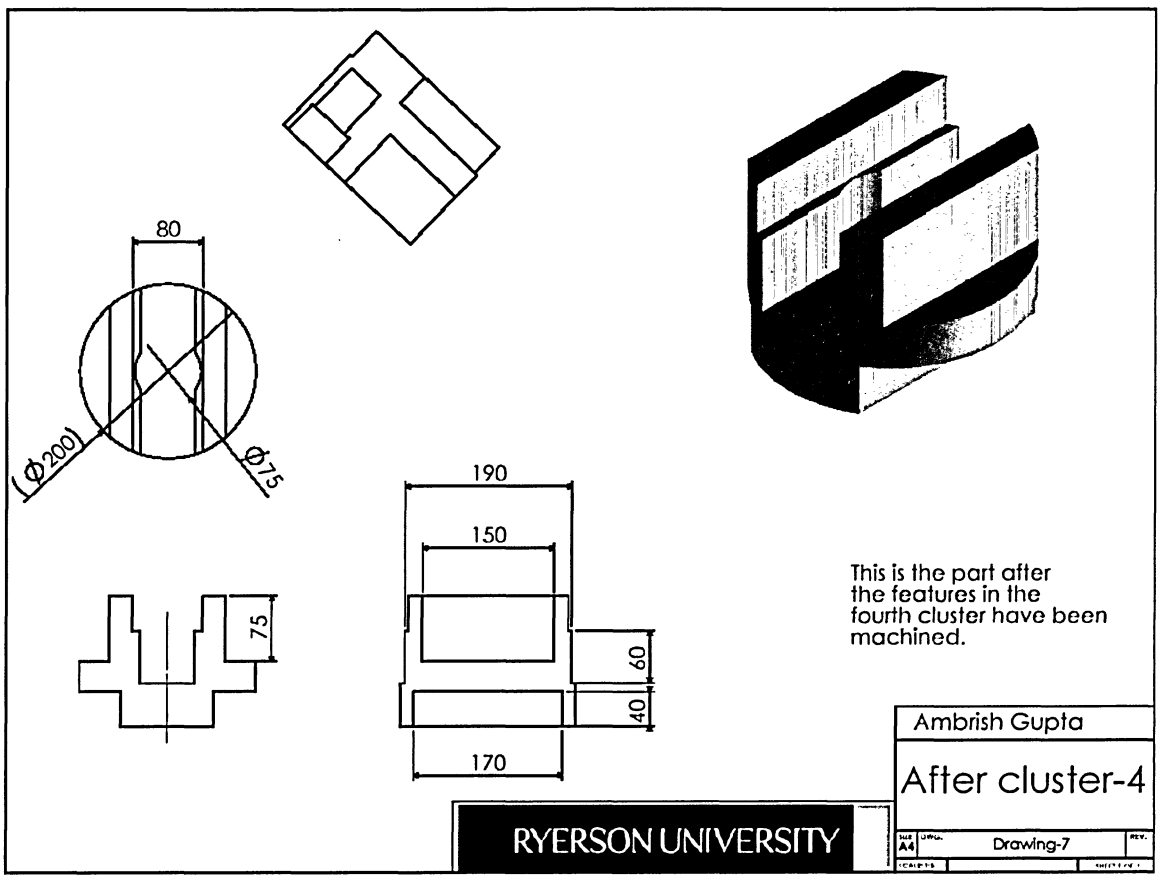


Figure 4.13: After machining feature-cluster-4

Figure 4.14 presents the part after Cluster-5 (C_5 = [*Holeblind1*, *Holeblind1a*]) has been machined.

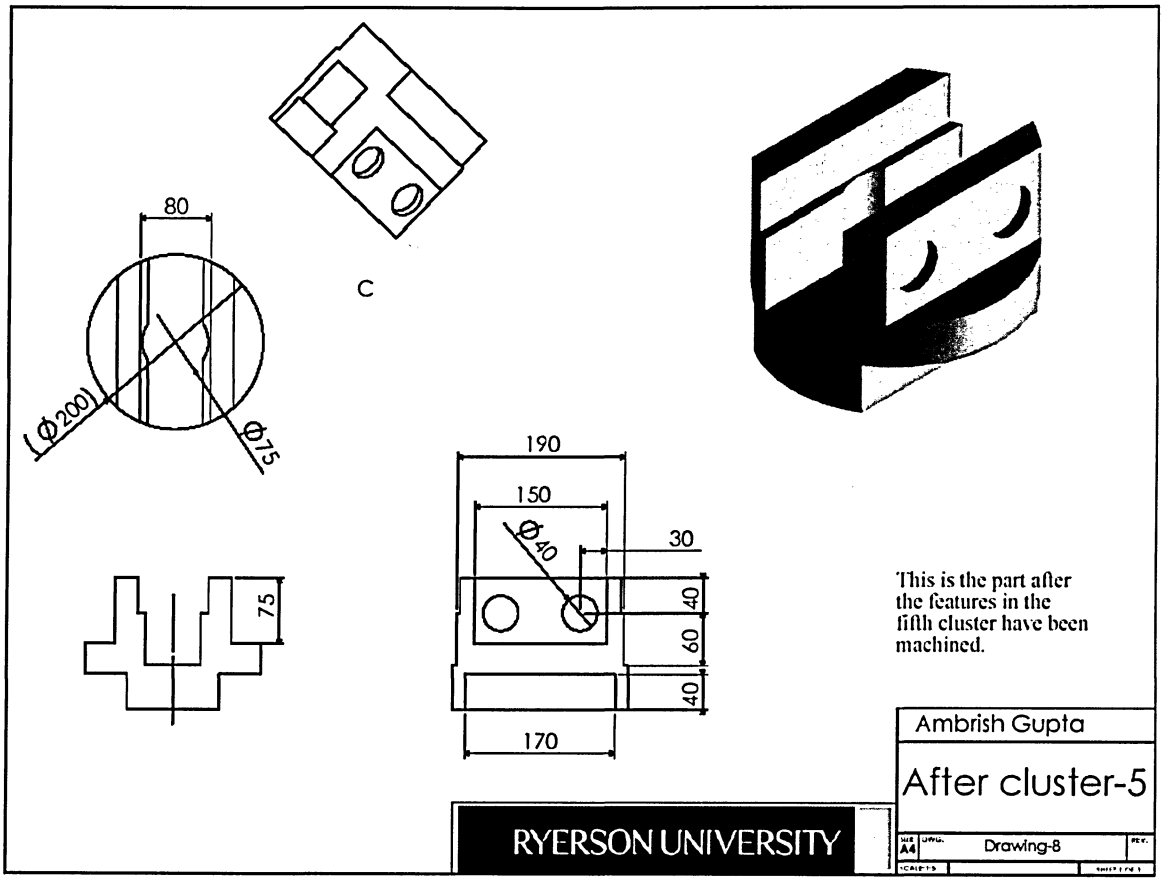


Figure 4.14: After machining feature-cluster-5

Figure 4.15 presents the part after Cluster-6 (C_6 = [*Holethru1*, *Holethru1a*],) has been machined.

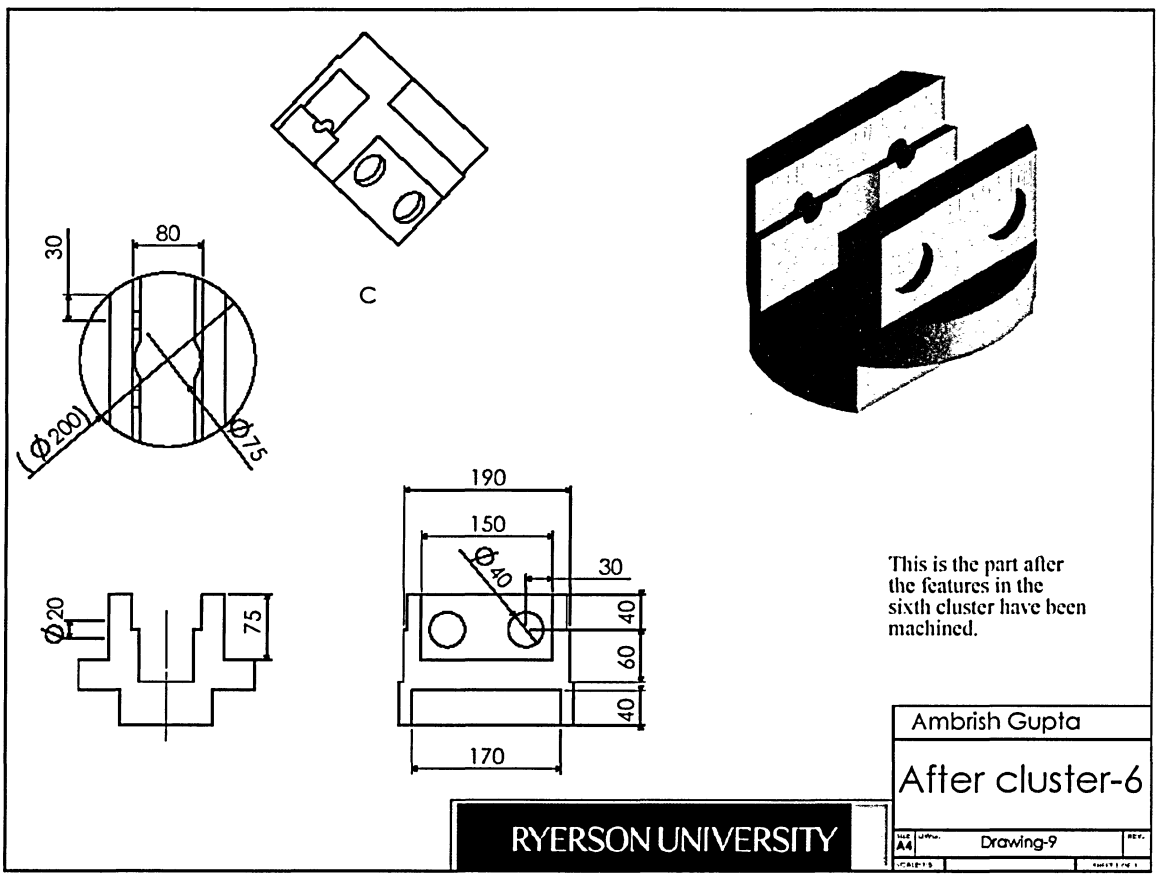


Figure 4.15: After machining feature-cluster-6

Figure 4.16 presents the part after Cluster-7 (C_7 = [*Holethru2.x*] $_{(x=1,...6)}$) has been machined.

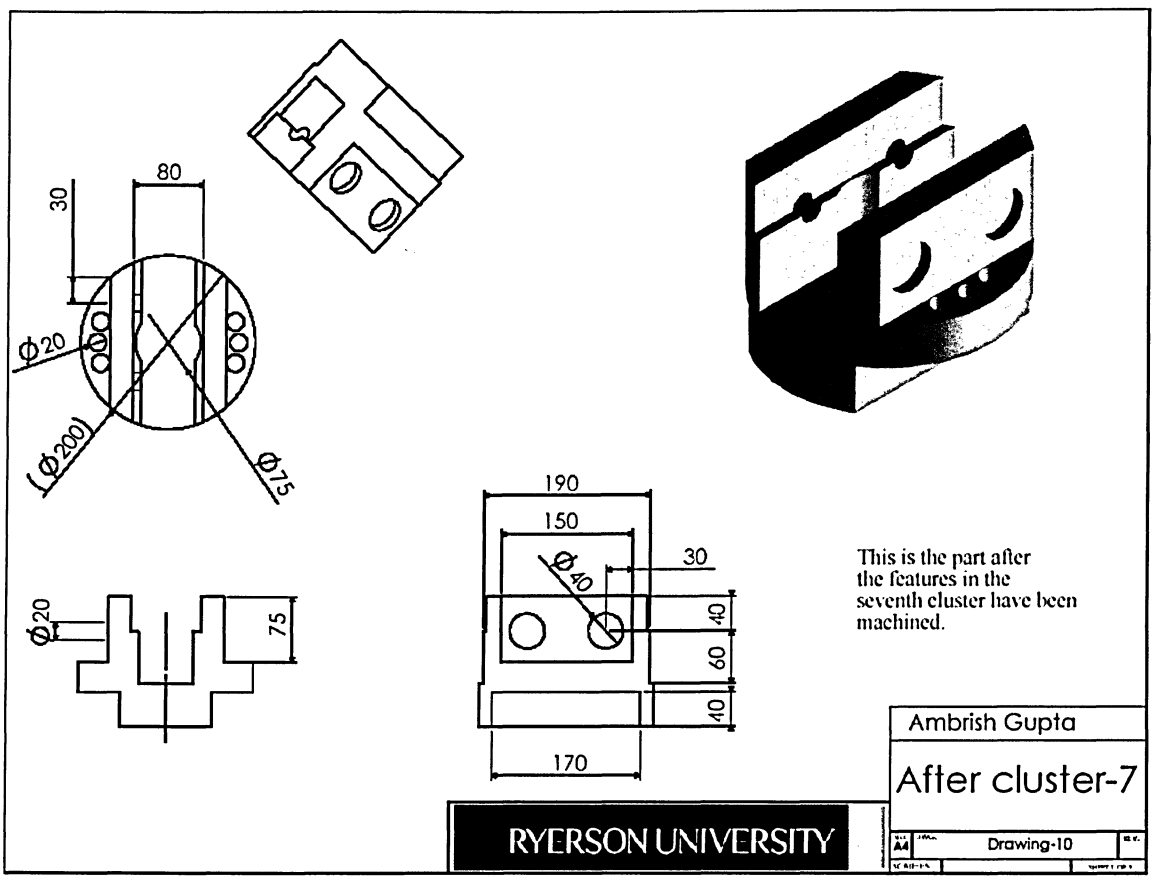


Figure 4.16: After machining feature-cluster-7

4.2.3.10. Discussion and Comments

An alternate method for clustering could be to partition a DSM of 'features vs. features' and then group features according to the TAD and rotation attribute. The drawback of this approach is that it gives a higher number of feature-cluster because it does not support discounting of weaker relations.

All the DSM operations performed by the DSM software (PSM32) are based on heuristics. Generally, heuristic-methods are a trade-off between the performance of the method and the computational expense. Thus, they do not guarantee correct results in all

conditions. In spite of the software being imprecise in some conditions, it works well for this research.

According to boolean-logic, any set of TADs resulting in any one of the prime-implicants be '1', constitutes the feasible TAD set for machining the complete part. On the other hand, the method given by Sarma and Wright [69], suggests determining the min-cover of the part (in terms of TADs). However, they do not consider the case where the min-cover has more than one implicants.

An intelligent method eliminates cycles by breaking a lower number of edges as compared to that with naive methods. A brief comparison of 'Intelligent cycle breaking' with their work, which identifies cycles using 'Depth-first search' and breaks the cycles naively (based solely on edge strength) is presented to demonstrate the benefit of intelligent cycle breaking.

The scenario represented by hypothetical graph and the corresponding hyper-graph presented in figures 4.5, and 4.6 respectively, is chosen to compare Sarma and Wright's method with the one developed in this thesis.

According to Sarma and Wright's [69] method:

- Conduct depth 1st search for cycle. Cycle = {C1, C2, C4, C1}.
- The weakest link the cycle: (C2-C4)
- Delete that link (because link is not "Critical").
- Conduct depth 1st search for cycle. Cycle = {C1, C2, C3, C4, C1}.
- The weakest link the cycle= (C1-C2)
- Delete that link (because link is not "Critical").

With the DSM based method ('intelligent cycle breaking'):

- The principal-circuit is: {C1, C2, C3, C4, C1}.
- When tearing advice is sought, the heuristic suggests breaking (C1-C2).

With only one edge being broken to eliminate the cycle as compared to two by using the Sarma and Wright's method shows the improvement over 'naive cycle breaking'. In a general problem also, by using the DSM method, there is a good possibility to delete a smaller number of hyper-edges to eliminate cycles.

***NB*:** In Sarma and Wright's work [69], they had a constraint that no strong relations be ignored until all the cycle causing weaker relations have been ignored. This constraint did not allow any room for intelligent cycle killing.

Similar to the past feature-clustering methods, current Computer Aided Process Planning (CAPP) systems also require input information on the manufacturing resources and the plant layout. However, in this thesis, features were clustered with an intent to design machine configurations. Therefore, a novel feature-clustering method was developed.

4.2.3.11. Summary

The method developed in this section inputs the part's feature-based model, and returns a set of feature-clusters. The clustering process consists of two phases. In the first phase, features are clustered based on their generic machining attributes. In this thesis, only TAD, tolerance, and machining precedence are considered. Since the machining precedence is considered, cycles may exist between the initial set of clusters. In the second phase, the cycles are removed by using a DSM based method. The output of this method is a set of feature-clusters. Each of these feature-clusters corresponds to a single RMT configuration. In the next section, a method is developed to determine the functional specifications of the RMT configurations required to machine the feature-clusters.

4.2.4. The Method for determining the specifications of modules

4.2.4.1. Introduction

Each of the feature-clusters resulting from the previous section corresponds to a single RMT configuration. The machining requirements posed by a feature-cluster are equivalent to the FRs of the corresponding RMT configuration. These FRs are manifested by the cooperation of the modules assembled into the RMT configuration. In principle, the FRs consist of motion requirements, velocity requirements, manipulability *etc.* The list of pertinent FRs is very large. To simplify, only the motion ranges are considered pertinent. However, the method developed in this section can be scaled-up to incorporate any number of independent FRs. Since this thesis considers the modules only in terms of their motions ranges, the modules are referred to as 'Motion Modules'.

In this section, a systematic method is developed to determine the functional specifications of the modules, which assemble to form the RMT configuration. The resulting RMT configurations are representative of an 'ideal machine,' specifically tuned for just that feature-cluster, i.e., The RMT is configured to render the machine, 'dedicated' or 'customized' for a particular feature-cluster (a task-set in general).

The prime intent the method being developed in this section is to determine the functional specifications of the modules, which can be feasibly assembled to machine the different feature-clusters. The goal of this method is to achieve functional feasibility. To achieve economic feasibility, the following sub-objectives are considered.

$$Max\left(\sum Utilization\left(\mathrm{RMT}_{C_i}\right)\right)$$

where RMT_{C_i} *is the RMT configuration for the feature-cluster- i*

This sub objective means that the total utilization of the RMTs be maximized. Each of these RMTs is specifically tuned for a specific feature-cluster. This objective is equivalent to minimizing the net redundancy for the RMT configurations, i.e., each RMT configuration embodies the minimal functions, and the net workspace of the RMT is

ideally just enough to manifest the required motions. This sub-objective is also analogous to maximizing the ROI.

$$Min\left(\sum \text{Re}\,configuation\left(RMT_{Ci} \rightarrow RMT_{Ci+1}\right)\right)$$

This sub-objective means that the total cost incurred while reconfiguring the machine to suit the succeeding feature-clusters be minimized , i.e., the total number of modules changed or rearranged, in order to reconfigure the machine from one configuration to the next, and so on is minimized. This sub-objective is equivalent to maximizing the Return On Change (ROC) from one RMT configuration to another. The best candidate that fulfils this sub-goal would be a RMT with all the functionality that the different feature-clusters require. This implies functional redundancy, which is in direct conflict with the previously stated sub-objective.

From the literature-survey, it was observed that Ye and Salustri [107], successfully solved a similar problem of simultaneously fulfilling two conflicting objectives. Therefore, the author chose to follow their approach and construct a consolidated objective. The objective for designing the RMT configurations was to co-optimize the stated sub-objectives, i.e. ,

$$Max\left(W_a \bullet \sum Utilization_{RMT_{C_i}} + \frac{W_b}{\sum \text{Re}\,configuration_{RMT_{C_i} \rightarrow RMT_{C_{i+1}}}}\right) \rightarrow \text{4-3}$$

Where $Utilization_{RMT_{C_i}}$ - is the 'Utilization', for the RMT corresponding to the feature-cluster-i.

$\text{Re}\,configuration_{RMT_{C_i} \rightarrow RMT_{C_{i+1}}}$ - is the 'Cost of Reconfiguring', for reconfiguring the RMT configuration corresponding to the feature-cluster C_i, to suit the succeeding feature-cluster (C_{i+1}).

W_a, W_b - are scalar weights, which correspond to the relative level-of-importance of the sub-goals.

Figure 4.17 shows the relative comparison of Dedicated, Flexible, and Reconfigurable machines with regards to the weights (W_a, W_b). If W_a=1, and W_b=0, the objective would be to design a dedicated machine. If on the other hand, W_a=0, and W_b=1, the objective would be to design a flexible machine. For designing reconfigurable machine, the weights should have to chosen somewhere in between.

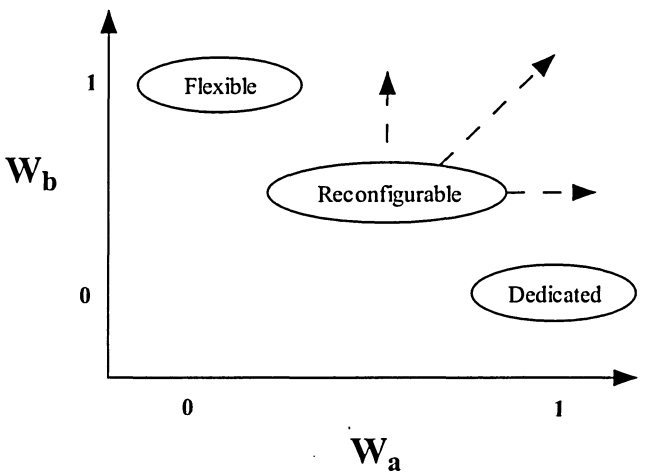


Figure 4.17: A different representation of Figure 1.1

NB: in this thesis, this objective (4.3) does not have a mathematical form, and is strictly abstract. Even though it does not serve as a mathematically defined objective-function, it serves as an abstract design objective, which the designer aims to achieve.

In order to simplify the objective and make it independent of the production-portfolio, the objective function is transformed into the following simplified objective.

Simplified Objective: Minimize the total number of modules. This contributes to minimizing the reconfiguration cost. Moreover, the redundancy in each RMT configuration ought to be minimized. This contributes to minimizing the wasted functionality or motion ranges.

This section presents a method to determine the specifications of the modules, which fulfill the afore-stated simplified objective. The method is then demonstrated by applying it to the example part. Finally, a discussion is presented, followed by a summary.

In reality, the modules could be functionally coupled. This renders the top-down strategy unsuitable, because it cannot decide which function to allocate to which module.

Therefore, as a simplifying assumption, the modules are assumed functionally independent, i.e., a module is assumed to contribute to a subset of motions in operation space which is exclusive to it. In mathematical terms, this implies that a module contributes to a set of rows of the Jacobian exclusive to itself. The Jacobian matrix is defined as the mapping of the manipulator's 'joint space' to its 'operation space'.

4.2.4.2. Approach

Figure 4.18 presents the approach followed to design the modules, which assemble to form RMT configuration required to machine a part.

Recall that this thesis follows a top-down design strategy. Inline with this strategy, firstly the machining requirements for each feature-cluster are identified. These machining requirements correspond to the FRs of the RMTs. The modules forming the RMTs cooperate to manifest the machining requirements.

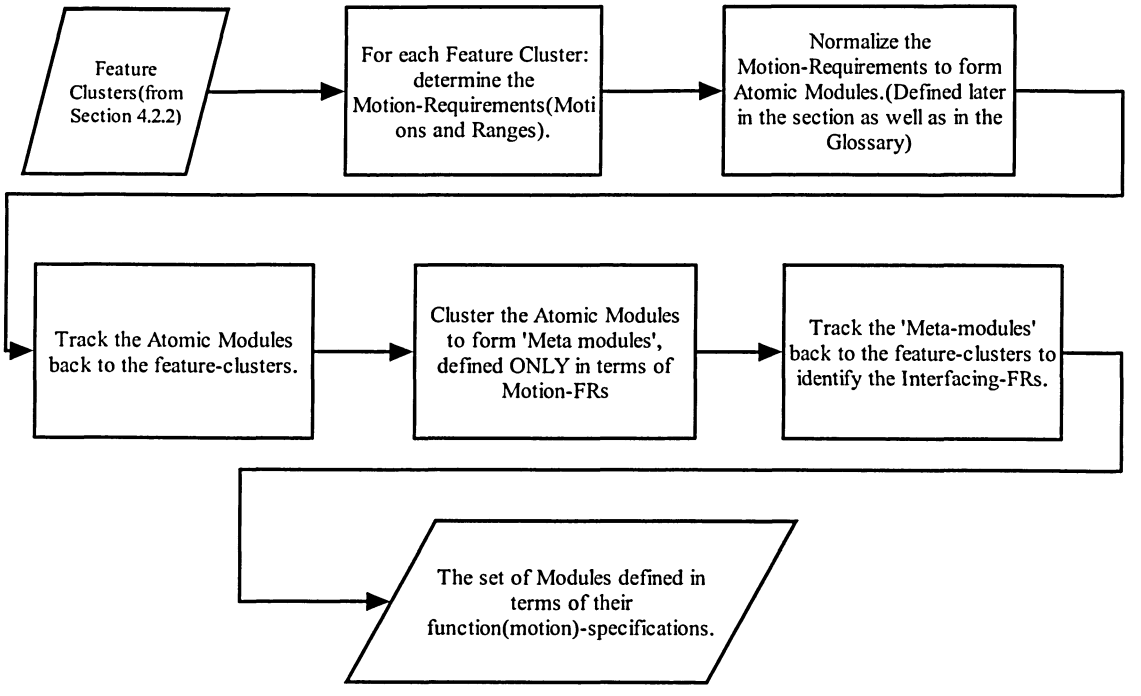


Figure 4.18: The approach for this section

The FRs are then normalized to form atomic modules. These modules are single DOF modules. The normalization process eliminates duplicates and contributes to fulfilling the

simplified objective stated in the previous sub-section. This process is analogous to Principal Component Analysis [31].

The resulting atomic modules are then backtracked to the feature-clusters. Based on this information, the atomic modules are clustered to form 'meta-module'. These meta-modules have more than one DOFs embedded. At this stage, the modules are defined only in terms of the motion-ranges they manifest. To define the module completely, it is necessary to specify the interfacing requirements.

To determine the interfacing needs, a suitable configuration is chosen for each RMT. This configuration is represented as a graph diagram. The existence of an edge between two functional-modules represents adjacency. This adjacency of modules dictates the interface design, i.e., they correspond to the interfacing requirements.

Based on the motion and interfacing requirements, the modules can be assigned physical detail. This task is highly iterative. In this thesis, it was proposed to use DSM to lay down the iteration scheme. However, because of time constraint, this scheme was not followed. Instead, the first reasonable physical design was chosen for each of the modules.

The final RMTs are formed by replacing the functionally defined modules in the configuration graph, by physically defined modules.

***NB*:** In this thesis, the focus is on determining the functional-specifications (and not detailed physical specifications) for the modules. The functional-specifications considered in this thesis are limited to 'Motion FRs' (the motions the modules manifest) and 'interfacing FRs' (the adjacency to other modules). In principle, the motion-FRs and interfacing FRs are coupled. To simplify the design task, the motion-FRs and interfacing-FRs are assumed un-coupled.

4.2.4.3. Step 1: Identify the Motion Requirements for each Feature-Cluster

The motion requirements for each feature-cluster are identified by analyzing the path the tool must follow in order to machine the feature-clusters. The first reasonable method to yield the required toolpath is used.

The toolpath is analyzed for both ranges of motions and coordination between Degrees-of-Freedom (DOFs). The motions are defined w.r.t. a coordinate-frame assigned to each

feature-cluster. This coordinate-frame is assumed to be the basis-frame for defining the RMT configuration's operation space.

Step1$_{\text{pre-processing}}$: Sequence the features within the feature-clusters according to precedence. In case of ambiguity, decide arbitrarily. The motivation to sequence the features is presented in the discussion on this step.

For each feature, choose a start-point. This is the point where the tool-tip is placed before it approaches the feature. This point is assumed to be on the part's bounding-box. The bounding box of a part is defined as the smallest cuboid that the part can fit in. In other words, it is an orthogonal-convex-hull of the part. The retract point is defined as the point where the tool-tip is placed after the feature has been machined. For the sake of simplicity, the start and retract points are assumed to be identical. The machining motions for a feature would include the motions required for the tool to approach the feature (from its start point), machine it, and return to the retract-point. Similarly, the positioning motion would be the motions required to place the tool-tip from the retract-point of one feature to the start-point of the succeeding one.

Assign a Right-handed coordinate frame to each feature-cluster, with its origin coincident with the start-point of the first feature of the feature-cluster. The frame's 'Z' axis is aligned with the TAD. 'X' and 'Y' are chosen to be an arbitrary pair of orthogonal axes, in the plane perpendicular to the TAD.

Step1$_{\text{Input}}$: Feature-clusters $\{C_i\}$ (from Clustering Phase).

Step1$_{\text{Process1}}$: Generate the motion requirements for machining the feature-clusters.

$\forall$ Feature-cluster:

$\forall f_i \in$ feature-cluster:

Identify the operation super-type. If tool is rotating, then the operation type is 'Milling' - which is a superset of Drill; else, it is 'Turning'. This classification of machining operations is consistent with STEP-NC (STEP-AP-238).

Generate the motions for tool-tip required to move the tool from the 'start' of the feature, machine the feature, and to return to the 'retract' point. Recall that the 'retract' point is

assumed coincident with the 'start' point. The tool-motions are generated by an external agent – either human or some suitable piece of software. Based on the machining motion, identify the motions, the kind of coordination between the DOFs, and the ranges of the motions.

NB: $\forall f_i$, the machining motions and parameters are specified w.r.t. its individual coordinate-frame.

After machining a feature, it has to position itself to the start-point of the succeeding feature. The motion from the 'retract point' of one feature to the 'start point' of the next is termed 'positioning motion'.

Find the positioning motion to move the tool from the 'retract-point' of 'f_i' to the 'start-point' of 'f_{i+1}' (the next feature), w.r.t. the coordinate-frame of the feature 'f_i'. This is done by joining the retract point of f_i to the start of f_{i+1} by a 3-D line; and if this line pierces through the bounding-box of the part, project it onto the bounding-box.

Based on the positioning motion, identify the motions, the kind of coordination between the DOFs and the ranges of the motions.

Transform the complete set of motions (for all the (sub) tasks in the feature-cluster) to represent it w.r.t. feature-cluster's coordinate-frame , i.e., the coordinate-frame of the first feature. This transformation operation is defined as the 'Union' of the motions. The

Step1$_{\text{Output}}$: Motions, their coordination, and their corresponding ranges (all specified w.r.t. the coordinate frame of the cluster's first feature).

Step1$_{\text{Discussion}}$: Ideally, to determine the toolpath, one needs to assume a conventional machine and identify/assume the operation(s) and tool. In the author's view, the best way to generate motions is to choose: (a) a Hyper-redundant robot (or a "Flexible Machine") with a tool mounted on its end-effector, and (b) a tool; and determine the path the Tool-tip needs to trace in order to machine the features. A hyper-redundant manipulator would have very high dexterity and would be able to reach points, which a non-redundant manipulator can possibly not reach. Moreover, the toolpath for such a manipulator would be smoother.

The toolpath and its parameters are dependent upon feature geometry, machining operation, tool's material, and geometry, part's material, surface finish, coolant flow and many other criteria. For simplicity, the motion parameters are calculated based on the part's material and the machining operation.

The bounding-box of the part changes after machining each feature-cluster. As a simplification, the bounding-box of the raw-stock is considered as the bounding-box for the part(s) at all its manufacturing stages.

Any suitable toolpath generation method can feasibly replace the method used here. Using commercial CAM software is a viable alternative. Its use has been precluded in this thesis because it requires considerable pre and post processing to yield the required motion ranges, and motion coordination. There is a strong potential to automate the CAM software to perform automatic pre, post processing, and yield the machining requirements in STEP-NC format.

The sequence of features does not affect the motion coordination or ranges. The main intent of sequencing features is to systemize the machining plan and to identify the first feature of the cluster. Recall that the first feature has a coordinate frame, which is the basis-frame for the RMT's operation space.

4.2.4.4. Step2: Normalize the Functional Requirements to yield Atomic Modules

In this step, the motion FRs (of the feature-clusters) determined from the previous step are normalized. Here, the normalization is performed by clustering the motions FRs. This is analogous to implementing Principal-Component-Analysis (PCA) [31] or Self-Organizing-Map (SOM) [39, 83] wherein, the dimensionality of the data set is reduced by clustering.

$Step2_{Input}$: The motion FR of the feature-clusters. These motions are defined w.r.t. the feature-cluster's coordinate frame.

$Step2_{Process1}$: The motion FRs are clustered using human intellect. To facilitate the designer, all the directional motion requirements from all the feature-clusters are listed together. Next, the designer clusters the individual directional motions intuitively. Finally, the "Union" over each cluster represents the corresponding 'normalized motion'.

The 'union' operator in this step is defined as the motion that can subsume all the other motions in the cluster, i.e., it is the Max of the values in that group.

$Step2_{Output}$: The set of normalized motions, each of which is manifested by single DOF modules, referred to as 'Atomic Modules.'

$Step2_{Discussion}$: The clustering operation is performed using human intellect. Even if a standard algorithm like PCA/SOM were used, the designer would still have a significant role in post-processing , i.e., deciding about the subsumption of one normalized motion by another. Since, no mathematical function has been developed for deciding upon the optimality of the module designs, human intellect is required to decide upon the subsumption of one module over another.

It is also possible that a DSM be tailored (as done by Bradley [3]) to include the effects of subsumption and perform intelligent clustering. This would eliminate any need for post-processing of information. Therefore, the whole clustering operation can be automated.

The directional motions are grouped separately even though they are physically the same , i.e., physically, each atomic-module is equivalent to a motor. There is a possibility that the motions be clustered after removing the directional context. The designer can choose any alternative without affecting the overall method.

Even though the velocity parameters of the motions are identified, for the sake of simplicity, they are not considered for the normalizing (clustering) operation. If the strategy were also to consider the velocities, the method would have to model each FR as a vector: <Motion range, Machining velocity range, Non-machining velocity range>, and then cluster these vectors. In this case, the 'union operator' would be the max of all the components such that the representative numbers (elements of the vector) cover all the entities in the cluster.

4.2.4.5. Step3: Cluster the Atomic Motions to yield meta-modules

This step is done with the intent of clustering the 'atomic modules' to form, 'Meta-modules'. Recall that these Meta-modules are a combination of atomic modules, i.e., they are multi-DOF modules.

Forming meta-modules would reduce the total number of modules required, and contribute to the fulfillment of the simplified objective stated in 4.2.4.1.

Step3$_{Assumption}$: In principle, a module can have a maximum of 6-DOFs. Having a 6-DOF module would lead to the possibility for the whole machine to be composed of a single module. This would reduce the granularity of the machine and would require a massive reconfiguration effort. Therefore, inline with Moon's assumption [55], it is assumed that a single module can have a maximum of 3 DOFs.

Step3$_{Input}$: (a) Feature-cluster's FRs and (b) Atomic Modules.

Step3$_{Process1}$: Cluster AMs to form Meta-Modules.

Backtrack the Atomic Modules (AMs), to the feature-clusters and represent this scenario as a 2-level DSM. The first level of the DSM is 'feature-clusters' versus 'feature-clusters'. Within each cluster, the relations between the AMs are 'Sequential' or 'Coordinated' (represented by '*S*' and '*C*' respectively); thus, making the second level of the DSM. A 'Sequential' relation between AMs implies that the motions must act one after another , i.e., not simultaneously. A 'Coordinated' relation between AMs implies that the motions they represent, must act simultaneously.

NB: This 2-level DSM is solely for representation only.

Next, the '*C*'s (coordinated relations) and '*S*' (sequential relations) are extracted separately from this 2-level-DSM. New DSMs are made for the '*C*'s and '*S*'s. These DSMs are AMs vs. AMs and the relations are boolean , i.e., they represent whether or not the AMs coexist in any feature-cluster. In other words, make a new DSM of "AMs vs. AMs." Wherever, two modules are used together in for machining a feature-cluster with an '*S*' relation, then there is a '0' , i.e., $Cell_{ij}$= '0' if M_i and M_j coexist for machining any feature-cluster.

A similar DSM is made for '*C*' relations.

Finally, both these DSMs are clustered (using standard DSM clustering algorithm; for details, Ref: [48]) to yield Meta-Modules. By definition, meta-modules embody more than one DOF. These DOFs could be either 'coordinated' (actuate simultaneously) or 'sequential' (actuate one after another). Although, the 'coordinated' module can subsume

a 'sequential' module, yet they are modeled separately. The rationale for modeling these two types separately is presented in Appendix-I.

Step3$_{\text{Post-Processing}}$: Some of the resulting meta-modules can possibly be subsumed by assembling others. However, the assumption made earlier of the modules being functionally independent leads to a constraint that no two assembling modules contribute to the same subset of operation space. In this step, the designer checks the resulting set of modules for subsumption. Having checked for subsumption, and eliminate the redundant ones to result in a set of modules are referred to as 'motion-modules'.

Step3$_{\text{Output}}$: Modules, which have been defined in terms of motion functions they need to perform in their operation space.

Each of these modules is represented as a 'Class' [43, 89] (in context to 'Object-Oriented-Programming').

Step3$_{\text{Discussion}}$: An explicit definition of the subsumption criteria is beyond the scope of this thesis. Therefore, the author chose to employ human intellect to decide whether to subsume a module.

The modules are represented as 'Classes'. This representation gives a structured information model of the module by explicitly specifying: (a) *attributes:* motion axes, motion speeds, mechanism being used *etc.* and (b) *functions/operations:* reconfiguration of channel for sequential modules, control functions, *etc.*

A class hierarchy is chosen (presented in Figure 4.19), wherein the 'generic'-module is modeled as an abstract-class. An abstract-class is defined as a class that cannot be instantiated to form objects. Rather, it acts as 'parent' to the classes that can be instantiated into an object [88]. This abstract class, represents the 'generic module, and is called the 'Module-Class.' The specific categories of modules derive from this abstract class. The specific types of modules inherit all the attributes and operations of the parent-class (the 'module' abstract-class). They impose some constraints on these inherited entities. E.g., the 2-D coordinated module has a specific type of mechanism. Moreover, some extra attributes or functions can be added to the derived class. Specific modules are

instances of these classes. These instances explicitly assign values to the fields of the corresponding class.

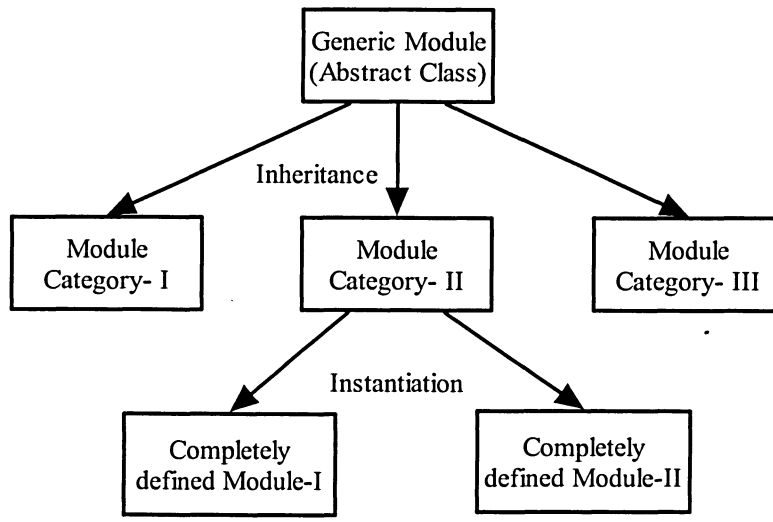


Figure 4.19: The class hierarchy

Figure 4.20 presents the class diagram of the module class (an abstract class). This class diagram is made in accordance with conventions recommended by STEP-AP-233(Systems Engineering). STEP-AP-233 is the extension of Unified Modeling Language (UML) [43], to represent the system at all the stages.

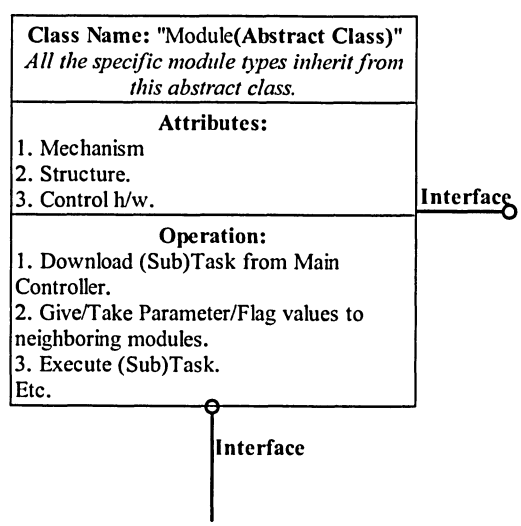


Figure 4.20: Class Diagram for a General Motion Module(Abstract Class)

Like other attributes of the class, the 'interface' for the modules is also modeled as a 'class'. This class has attributes only. STEP-AP-233 (Systems Engineering) recommends this convention. The class diagram for the interface is presented in figure 4.21.

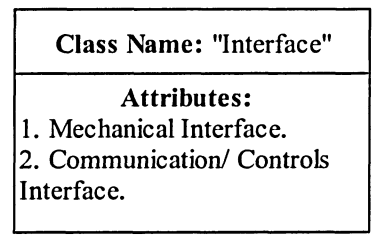


Figure 4.21: The class diagram of the 'Interfaces'

The benefits of modeling 'modules' as classes are:

- This representation gives a structured functional description of the module; thus, facilitating the process of detailed design.
- It represents the module as a structured information model, which facilitates its use through computer-applications. These computer-applications could search the World Wide Web for appropriate pre-designed modules. Alternatively, a computer-application could also be developed to design the modules automatically.
- This representation is complaint with STEP-AP-233. Thus, this representation facilitates the use of STEP-based software applications.
- The class diagrams, the inheritance and implementation models can help modularize the modules to a higher degree of granularity. E.g., The sequential and coordinated 2-D modules have everything in common except that the sequential one has an extra attribute and related functions. Therefore, the class diagram gives a clear picture that these two classes can be modularized: there be one generic '2-D module' of a particular position and velocity range; it would be made 'Sequential' or 'Coordinated' by adding an attachment; depending upon the attachment, the module behaves as Coordinated/Sequential.

***NB*:** In reality, there are a large number of attributes and operations for the modules. This thesis attempts to present a very limited set of them. However, the scheme developed here can be extended to incorporate any number of attributes or operations.

4.2.4.6. Application of the method to the example part to yield functionally defined modules

Figure 4.22 presents the toolpath for machining the feature-cluster-6. The machining motions, and positioning motions are shown.

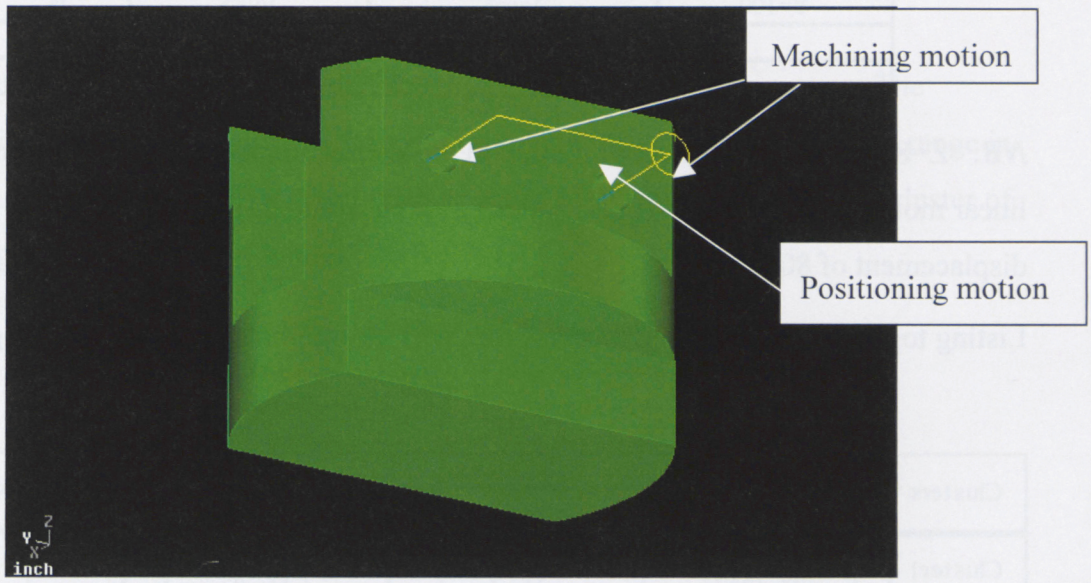


Figure 4.22: The motions required for machining feature-cluster 6 are shown

Machining Operation: **Drilling** (a subset of milling operations (Step-NC)).

The machining, and positioning motions are analyzed to result in Table 4-9.

Table 4-9: The results from the FR determination step for feature-cluster 6

Machining						**Non-Machining**
Serial #	*Feature*	*Operation super type*	*machining motion*	*machining speeds*	*motor KW*	*positioning motion*
1	holethru1	milling	z=80	4	25	positioning F#1 to F#2 : y=100
2	holethru1a	milling	z=80	4	25	

Feature-cluster-6 was chosen for the ease of demonstration. The machining speeds are determined from the machining handbook, Ref: [87]. The length units are 'mm' and speed units are 'mm/sec'.

The unionized motions (all net motion requirements represented in the coordinate-frame of the cluster's first feature) for feature-cluster-6 are presented in Table 4-10.

Table 4-10: List of the motions requirements for Feature-Cluster

Motions			Spindle Motor KW
Ranges	machining vel	non-machining vel	
Z=80	Vz=4	Vz=8	
Y=100	Vy=3	Vy=8	25
Co-ordination = NULL			

***NB*:** "Z=80" means that range of motion in the Z direction is 80mm. This means that a linear motion module, which provides this motion would have the maximum displacement of 80 mm.

Listing together the unionized motions for every feature-cluster, Table 4-11 is gotten.

Table 4-11: The list of FRs for all the feature-clusters

Clusters	Motions			Spindle KW
	ranges	*machining vel*	*non-machining vel*	
Cluster1	Z=180	Vz=4	Vz=8	18
	X=120	Vx=4	Vx=8	
Cluster2	Z=20	Vz=4	Vz=8	18
	X=120	Vx=3	Vx=8	
Cluster3	Z=150	Vz=4	Vz=8	30
	X=210	Vx=3	Vx=8	
	Y=210	Vy=3	Vy=8	
	Coordination {X,Y}			
Cluster4	Z=60	Vz=4	Vz=8	30
	X=210	Vx=3	Vx=8	
	Y=210	Vy=3	Vy=8	
Cluster5	Z=25	Vz=4	Vz=8	25
	Y=100	Vy=3	Vy=8	
Cluster6	Z=80	Vz=4	Vz=8	25
	Y=100	Vy=3	Vy=8	
Cluster7	Z=130	Vz=4	Vz=8	25
	Y=180	Vy=3	Vy=8	
	X=180	Vx=3	Vx=8	

Listing the motion requirements together (with no reference to the feature clusters) results Table 4-12.

Table 4-12: Listing all the class's requirements together

Direction	Ranges						
Z	20	25	60	80	130	150	180
X	120	180	210	Null			
Y	120	180	210				
Motor KW	18	25	30	Null			

Motions listed in Table 4-12 are normalized, by clustering the motion ranges. The representative value of each cluster of the motion range is the maximum motion range in that cluster. The process of identifying the maximum of the motion ranges in a cluster of motion ranges is referred to as 'union'.

Table 4-13: Normalization of the motions

Tag	Ranges				Union
Z1	20	25	60	80	80
Z2	130	150	180	Null	180
X1	120	180	210		210
Y1	120	180	210		210
Spindle-1	18	Null			18
Spindle-2	25	30	Null		30

Taking union over it the result is z_1=80; z_2=180; x=210; y=210; Motor KW = 18, 30;

Therefore, we get (considering only the motions).

AM1 → "z=80"; $V_{m/cing}$ = 4mm/sec; $V_{non\text{-}m/cing}$ = 8 mm/sec;.

AM2 → "z=180"; $V_{m/cing}$ = 4mm/sec; $V_{non\text{-}m/cing}$ = 8mm/sec

AM3 → "x=210"; $V_{m/cing}$ = 4mm/sec; $V_{non\text{-}m/cing}$ = 8mm/sec

AM4 → "y=210" ; $V_{m/cing}$ = 4mm/sec ; $V_{non\text{-}m/cing}$ = 8mm/sec

The atomic motion modules are backtracked to the feature-clusters. The 'S' (in the DSM cells) represents sequential motions of the atomic modules (they act one after another). The 'C' (in the DSM cells) represents coordinated motions of the atomic modules (they actuate simultaneously).

In Table 4-14, ∀ feature-cluster, the Atomic modules used, and the relations between them are identified and corresponding marks are entered in the cells. 'S' = sequential; 'C' = coupled.

Table 4-14: DSM showing all the setups

	C1	C2	C3	C4	C5	C6	C7
C1	2 * S 3 S *						
C2		1 * S 3 S *					
C3			2 * S 3 S * C 4 C *				
C4				1 * S 3 S * 4 S *			
C5					1 * S 4 S *		
C6						1 * S 4 S *	
C7							2 * S 3 S * 4 S *

There is only one pair of AMs related by the 'C' relation. Therefore, there is no need to make a DSM of AMs with 'C' relations. The 'C' relations yield an 'initial-Meta-Module', $\{AM_3, AM_4\}$[8] .

[8] The {} brackets have a special significance: The atomic-motions enclosed in the brackets are "Coordinated" i.e. the motions are simultaneous.

Making a DSM of AMs vs. AMs, such that M_{ij}= '0' if M_i and M_j have 'S' relation in any feature-cluster, Table 4-15 is gotten.

Table 4-15:DSM of AMs vs. AMs

	AM1	AM2	AM3	AM4
AM1			0	0
AM2			0	0
AM3	0	0		0
AM4	0	0	0	

Upon clustering, the DSM suggests a meta-module formed from AM_3 and AM_4, and AM_1 and AM_2 each form a single cluster.

Following the clustering from the DSM, the initial set of modules is presented in the Table 4.16.

Table 4-16: Final lists of Motion Modules

	Coupled	Sequential	Independent
1	$\{AM_3,AM_4\}$	(AM3,AM3)	AM_1
2			AM_2
Union	$MM_1=\{AM_3,AM_4\}$	$MM_2=(AM_3,AM_4)$	$MM_3=AM_1$; $MM_4=AM_2$

The modules are not subsumed by other modules and this was the decision of the designer (in this case, the author). Therefore, the final set of modules is as follows:

FR for Module#1: Coupled motions in 2 orthogonal directions. Ranges of both orthogonal motions = 210 mm. machining and non-machining speed in both directions are 4 and 8 mm/s respectively.

FR for Module#2: Sequential motions in 2 orthogonal directions. Ranges of orthogonal motions = 210 mm. machining and non-machining speed in both directions are 4 and 8 mm/s respectively.

FR for Module#3: Single motions. Range of motion = 80 mm. Machining and non-machining speed are 4 and 8 mm/s respectively.

FR for Module#4: Single motions. Range of motion = 180 mm. Machining and non-machining speed are 4 and 8 mm/s respectively.

Figure 4.23, 4.24, and 4.25 present the class diagrams for the 2-D sequential module, 2-D coordinated module, and 1-D module respectively.

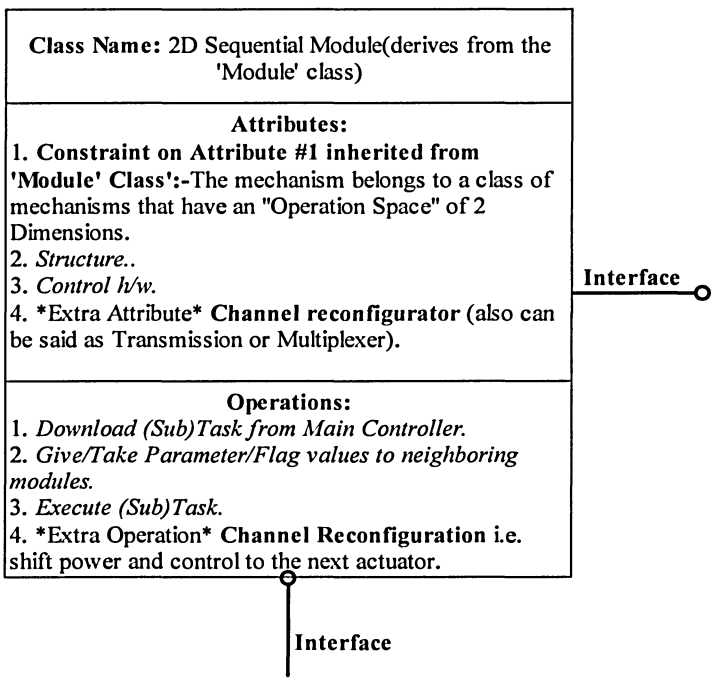


Figure 4.23: Class Diagram for a 2-D sequential Motion Module

***NB*:** All the 'Module' classes derive from the 'Module' abstract class. The derived classes inherit the attributes and operations of the base class. The constraints, additional attributes, or operations in the derived classes are mentioned.

For example, the 2-D sequential module derives from the Module-Class and has the following attributes in addition to those inherited from the base-class:

- Specification that the mechanism should have a 2-DOF operation space.
- An extra attribute to act as a "Channel Reconfigurator," which essentially is a multiplexer/transmission.
- An extra operation to govern the action of the "Channel Reconfigurator."

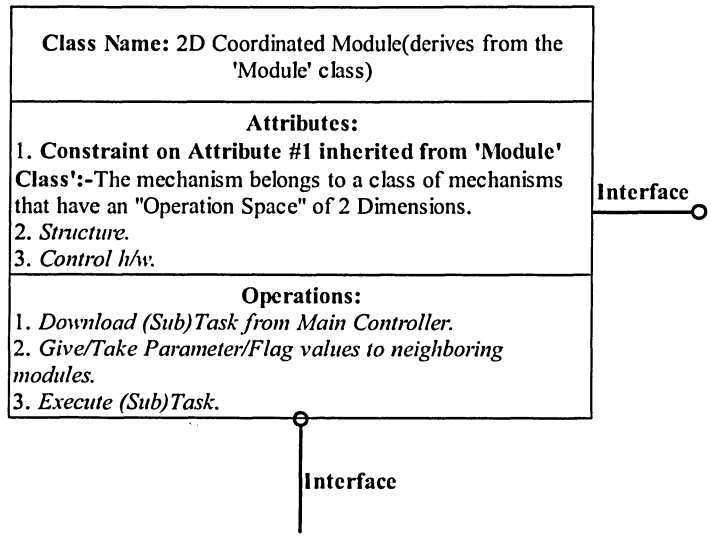


Figure 4.24: Class Diagram for a 2-D coordinated Motion Module

When considering coupled motion manifested by 'Coordinated motion modules', the degree of interdependence (communication) between the motions may potentially affect the clock frequency of the microprocessor, the bandwidth requirements of the internal and external communication bus *etc.*

There is a strong possibility to use a non-Cartesian mechanism, because the DOFs actuate simultaneously. Using such a mechanism would complicate the mechanics and controls. However, as compared to cartesian mechanisms, they typically require actuators of smaller capacities because typically non-cartesian mechanisms have high amplification ratios.

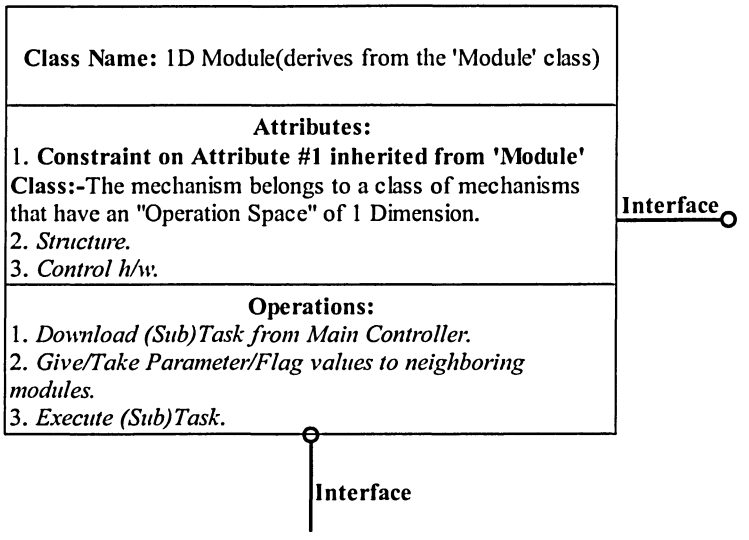


Figure 4.25: 1D Motion Module

***NB*:** These Class diagrams are strictly conceptual representations of module categories. Moreover, they have not yet been embedded with the motion and velocity information. These module classes are instantiated to form objects. These objects would encapsulate the motion and velocity information, and be the final information module of the modules. Based on this information model the detailed design can be performed.

4.2.4.7. Step 4: Identify the 'Interfacing requirements'

In the previous section, the motion FRs for the modules were determined. However, to have a complete functional specification of the modules, it is necessary to identify their interfacing requirements. In this section, the interfacing FRs are identified.

Assembling the physical modules (which have not yet been developed in this thesis) to yield appropriate RMT configurations is usually referred to as 'configuration design' [55]. The modules have to be designed so that they can be arranged in suitable configurations, and the configurations must be designed so that the modules can be feasibly assembled to fulfill the designated tasks. Therefore, the 'module-design task' and the 'configuration-design task' are coupled. The ideal solution would be to design both the modules and the configurations simultaneously. To simplify this, the functional description of the modules is determined first; next, a feasible configuration is chosen out of the many possible ones. The configuration is represented as a graph, and this graph specifies the interfacing-FRs

of the modules. The interfacing-FRs considered in this thesis are limited to only to the adjacency of modules. The adjacency requirements are required as an input to the detailed design step. In the detailed design process, modules are assigned physical detail.

A machine is analogous to a pair of cooperating robotic-manipulators. Extrapolating this fact into the RMT domain leads to two chains of modules: (a) base to tool and (b) base to work-piece. Defining the functional-configuration is analogous to assigning modules to these chains. "To minimize the errors, the total number of DOFs are distributed equally to these chains," [55]. Therefore, the modules are assigned to the two chains such that the total number of DOFs are distributed evenly.

$Step4_{Input}$: The feature-clusters and modules defined in terms of the motions FRs.

$Step4_{Process}$: Assign motion modules to the two chains, such that the total number of DOFs is distributed evenly on the arms.

Figure 4.26 shows the configuration diagram of the RMT configuration. The configuration is analogous to two chains of modules. The links between the modules can be populated with structural modules, which provide stiffness to the RMT. At this stage, the structural modules have not been designed. A detailed discussion on the design of structural-modules is presented in section 4.2.4.8.

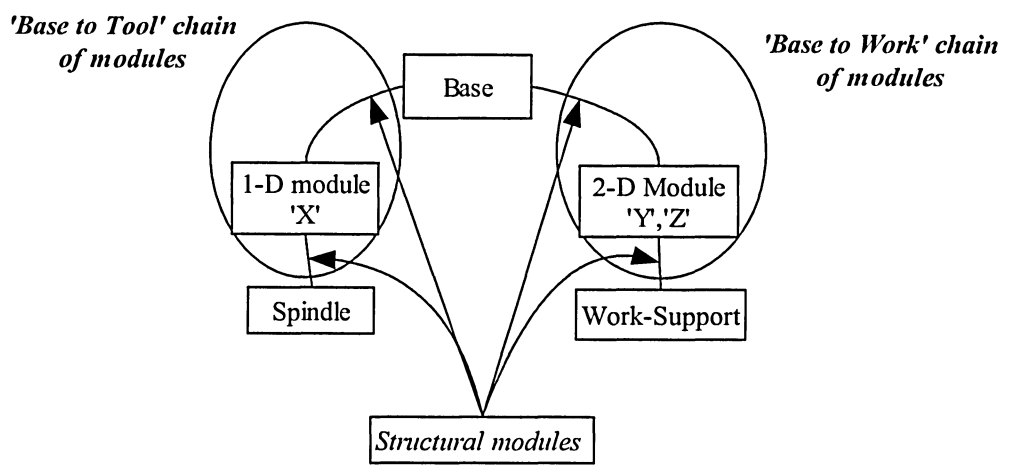


Figure 4.26: Configuration Diagram (for Feature-cluster 4)

Step4$_{Output}$: The interfacing requirements of the motion modules. At the end of this step, the modules have a complete functional description, and can proceed to the step of detailed design.

Step4$_{Discussion}$: The toolpath gives the relative motion between the tool and the work-piece, i.e., the toolpath does not give any information about the 'individual motions' of the tool or work, i.e., the toolpath does not suggest anything about the DOFs assigned on the chains. Therefore, minimization of error is chosen to be the only significant criterion. Distributing the DOFs evenly on the two arms reduces the error stack-up, i.e., it contributes to minimizing the overall error of the RMT configuration. Therefore, this objective holds a significant importance for choosing a feasible RMT configuration.

The topologies are highly abstract , i.e., they only show the modules on each arm. Even so, following the criteria of evenly distributing the DOFs to the chains, the number of possible configurations is more than one. In the ideal case, the different possibilities should be evaluated based on heuristics. For simplicity, the first reasonable configuration is chosen.

There may be any number of structural modules embedded between two motion modules. This is not captured in the configuration diagram. Therefore, the configuration diagram does not strictly represent the adjacency between modules.

The topologies of RMT configurations affect the detailed design of the modules. A module is possibly used in many RMT configurations; thus, it should be suitable for all the corresponding configurations. The following step of detailed design would involve iterating over the feature-clusters, where the module is used.

4.2.4.8. Step 5: Design modules

The previous step resulted in the complete functional specifications of the modules. To demonstrate the feasibility of the method, it was important to assign physical detail to these functionally defined modules. The modules participate in more than one RMT configurations. Therefore, for the modules to feasibly assemble in all the required RMT configurations there is an inherent need for iteration. This is a very complicated process. However, considering the timeframe for this thesis, the detailed design of the modules

and the RMTs was chosen to be beyond the scope. Therefore, the first reasonable physical-design was chosen for these functionally defined modules.

$Step5_{Input}$: Feature-clusters, functional modules, and the configuration diagrams.

$Step5_{Process}$: Choose the first reasonable physical design for the modules (defined in terms of their FRs) such that the resulting modules feasibly assemble in the required RMT configurations. Moreover, the implicit function requirements of the resulting RMTs in terms of error, manipulability, reach *etc.* should be fulfilled by the assembly of the resulting RMT configurations.

$Step5_{Output}$: Completely defined physical modules.

$Step5_{Discussion}$: The task is to choose the designs for the modules, such that each module satisfies its corresponding motion-FRs and interfacing-FRs. Since the modules have to be feasibly assembled in all the required RMT configurations, the detailed design of the modules would be an iterative process. Moreover, the RMT configurations thus formed should have adequate manipulability, error characteristics, structural stiffness, thermal compensation, *etc.* This further strengthens the need for iterations.

In addition to the motion modules, the RMT configurations also require structural modules. These modules contribute to the RMT's stiffness. Designs of the structural modules are coupled with the designs of the motion-modules. The same argument applies to the design of material-handling system (MHS). The MHS performs the task of fixturing the part in the proper pose (w.r.t. the RMT). Since their designs are coupled, designing them implies iterating until convergence. From the review on DSM (presented in Chapter 4), it was observed that DSM has been successful application in a range of similar design problems [53, 48]. Therefore, DSM was used to lay the iteration-scheme.

Design of each module is analogous to assigning values to a parameter lump. Modeling the functional modules as classes makes this analogy even more reasonable.

This DSM shows the dependence of the modules on each other. The modules depend on others only if they participate in the same RMT configuration. The participation of the motion modules in different RMT configurations is presented in Table 4-17. In this table, $Cell_{ij}$=0, if RMT for feature-cluster- i, requires $Motion\text{-}Module_j$.

Table 4-17 also gives a quick view of the motion modules used in each RMT configuration. The information presented in this table, also helps the designer to construct the DSM presented in Table 4-18. Recall that the design of a particular module affects that of another only if the two modules coexist in a RMT configuration. Therefore, in this DSM, $Cell_{ij}$ =0, if $Module_i$ and $Module_j$ coexist in a RMT configuration.

Table 4-17: Final modules used in each setup

	MM_1	MM_2	MM_3	MM_4
$RMTC_1$		0		
$RMTC_2$		0		
$RMTC_3$	0			0
$RMTC_4$		0	0	
$RMTC_5$		0		
$RMTC_6$		0		
$RMTC_7$		0		0

Table 4-18: Partitioned DSM. The different partitions are shown

		Motion Modules				Structural Modules	Material Handling system.
		MM1	**MM3**	**MM2**	**MM4**	**S**	**MHS**
Motion Modules	**MM1**	*	0			0	0
	MM3	0	*	0		0	0
	MM2		0	*	0	0	0
	MM4			0	*	0	0
Structural Modules	**S**	0	0	0	0	*	0
Material Handling system.	**MHS**	0	0	0	0	0	*

Upon partitioning, this DSM (Table 4-18) suggests three interlinked partitions ([MM_1, MM_3], [MM_3, MM_2], [MM_2, MM_4]) and an integration partition ([S, MHS]). The structural modules, and the material–handling system act as the integrating entities, i.e., all the motion-modules have a coupling with the structural modules and the material–handling system. A design change to any module propagates to all the other modules

through the integration partition. This DSM demonstrates the complexity of the design task.

To reduce the iterations, the DSM is torn. It is assumed that firstly, the designs for the motion modules evolve. Once they converge, the structural modules and the material handling (sub) systems are designed iteratively, while considering the design of the motion modules 'Frozen'. The structural modules and the material handling system are designed to result in a RMT configuration such that it: (a) allows the part to be mounted (by the material-handling system) in the aligned position and (b) does not fall apart under the machining loads generated while manifesting the assigned motions (all of which are defined w.r.t. the feature-cluster's origin).

In Table 4-19, the partitions are shown in different colors and are enclosed by different border types. The torn DSM can be interpreted in a variety of ways. The following paragraph presents one of the possible interpretations.

Table 4-19: The torn DSM

		Motion Modules				Structural Modules	Material Handling system.
		MM1	**MM3**	**MM2**	**MM4**	**S**	**MHS**
Motion Modules	**MM1**	*	0			2	2
	MM3	0	*	0		2	2
	MM2		0	*	0	2	2
	MM4			0	*	2	2
Structural Modules	**S**	2	2	2	2	*	2
Material Handling system.	**MHS**	2	2	2	2	2	*

The design scheme resulting from the DSM is presented as follows:

- Design M_2.
- Based on it, design M_3 and M_4.
- Based on M_4's design, design M_1.

- Considering the current design, and M_1 and M_2 frozen, re-design M_4.
- The new design of M_4 and M_3 affect the design of M2. Thus, based on the current designs of M_3 and M_4, M_2 is re-designed.
- Until the designs converge, goto step #2.
- With the design of the motion module 'frozen', iteratively design the structural modules, and the material-handling system until their designs converge.

Once the designs for the motion modules have matured, the structure and the material handling systems are designed, while considering the converged designs of the motion modules 'frozen'.

To design the structural modules, firstly, motion-modules are placed in free-space such that they can manifest all the required motions. Next, a structure is made as a 'harness' to hold the motion-modules in place. This process is executed for all the RMT configurations. The harnesses for all the different RMT configurations are modularized. However, the main problem posed by this approach is that one can have infinite number of possible shapes for the harnesses. In this thesis, the first reasonable structure is used.

The DSMs presented in Tables 4.17, and 4.18 were developed with an intent to lay down the design scheme. However, inline with Helo's work [24], a DSM can be used to lay down a scheme to choose a set of pre-designed modules, which fulfill the functional definitions of the modules. This DSM would be a DSM of 'Decisions' for module designs , i.e., decisions v. decisions.

Considering the fact that designing modules requires multi-disciplinary specialization, automating the design process would increase the feasibility of the overall method developed in this thesis. The class representation being a structured information model supports its use by an autonomous software agent. This agent would assign physical details to the functionally defined modules. The agent can even be configured to search the knowledge repository from the WWW for appropriate pre-designed, off-the shelf modules or components. The following paragraphs present a proposal for developing an 'autonomous design agent'. This agent would integrate a whole range of virtual product

development tools via an Active-X application. Active-X is a set of "strategic" object-oriented programming technologies and tools.

The motion module could be objectified as follows:

- The 'mechanism sub-class' could be an empty MSC.ADAMS[9] command file, or an empty SimMechanics[10] model.
- The communication and control system could be an empty Simulink[11] model. This model would encapsulate the control elements (both continuous and finite state), and the communication system.
- The 'physical-structure' would be represented as an empty string of 'Shape Alphabets' [4, 19]. This approach is proposed to be implemented by using SolidWorks.

All these sub-classes can be integrated using Application Protocol Interface (API) of the software-packages into a single class representing the module.

The autonomous design agents would fill in the empty files/models/strings (possibly by using an evolutionary algorithm) to define the subsystems making the module, hence the module.

The design agent follows the evolution scheme and conducts incremental iterations, i.e., the design for all a module updates on the basis of its current design state, and the current design stated of its adjacent modules.

As an alternate to DSM based incremental iteration, it is possible to update all the module designs in parallel. This scenario is similar to a 'Cellular-Automata' (CA). Typically, the cells of the CA are 'Finite State Machines' (FSMs). Although, in this case, the architecture is similar to a CA, yet the cell should have computation power, memory, and communication interfaces. Some researchers refer such kind of cells as 'X-Machines'. "The X-machines are more expressive and flexible than FSMs" [37]. Thus, the author

---

[9] Ref: <www.mscsoftware.com/adams>
[10] Ref: <www.mathworks.com/simmechanics>
[11] Ref: <www.mathworks.com/simulink>

proposes a CA based network, with each of the cells being an X-Machine. Using a CA based design process, requires large computational resources. Therefore, the author recommends using parallel processing. With the advent of 'Hyper-Threading' enabled processor, it is possible to implement parallel processing on a single computer.

4.2.4.9. Step 6: Form RMT configurations.

In this step, the functional modules in the configuration diagram of the RMTs are replaced by the corresponding physical modules.

$Step6_{Input}$: Configuration diagram and the Physical modules.

$Step6_{Process}$: Replace the functional modules in the configuration diagram of the RMTs with the corresponding physical modules.

$Step6_{Output}$: The RMT configurations for each feature-cluster.

4.2.4.10. Application of the method to the example part to yield the final RMT configurations

The method was applied to the example part (Figure 4.7), and the resulting RMTs are presented in the Figures 4.27 – 4.31. In these figures, the motion-modules and the structural modules are labeled.

***NB*:** The following figures (RMT diagrams) do not show the couplings.

Figure 4.27 shows the RMT configuration required to machine feature-cluster-1 and feature-cluster-2. This RMT configuration is suitable for turning machining function, i.e., the work rotates. This configuration has 2-DOFs; however only one of these is active at a time.

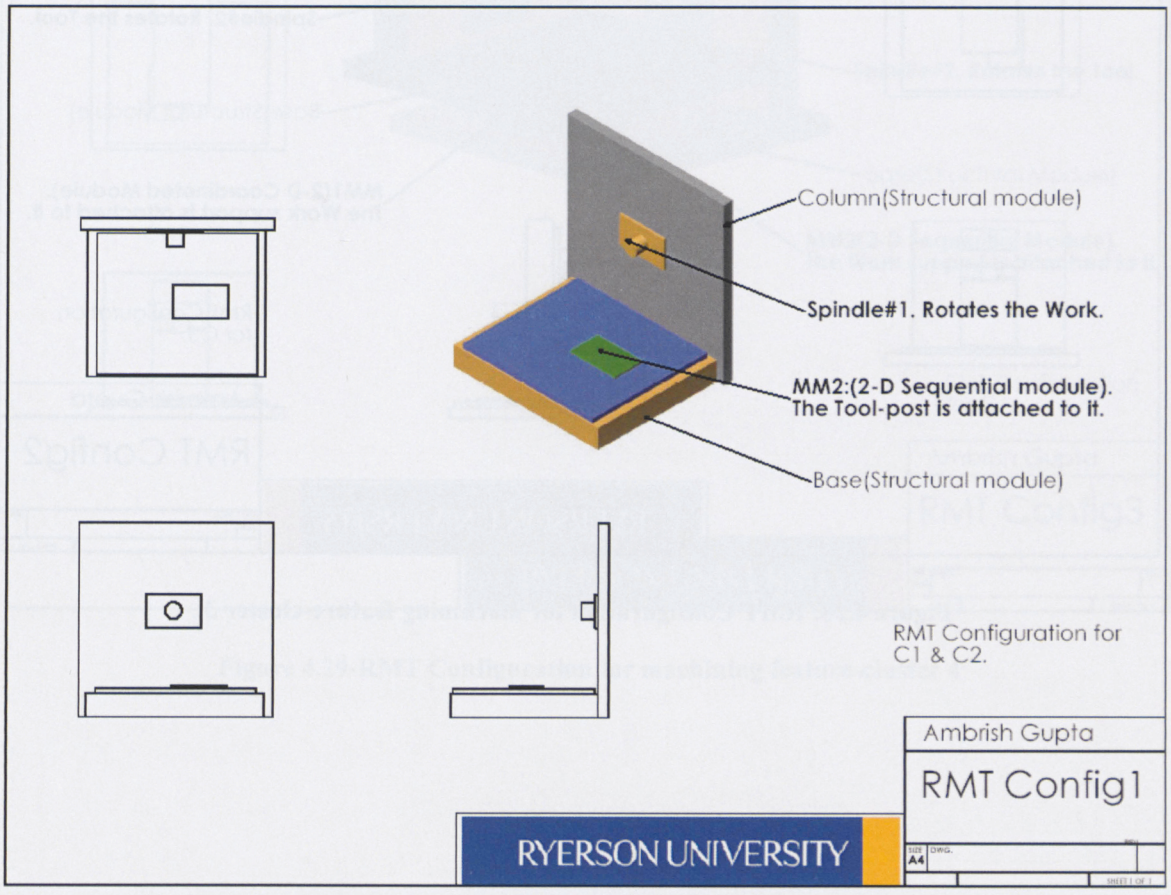


Figure 4.27: RMT Configuration for machining feature-cluster 1,2

Figure 4.28 presents the RMT configuration required to machine feature-cluster 3. This RMT configuration has 3-DOFs; however, maximum of two of these DOFs are active at a time. The RMT configuration requires the tool to rotate, thus it performs the milling machining function.

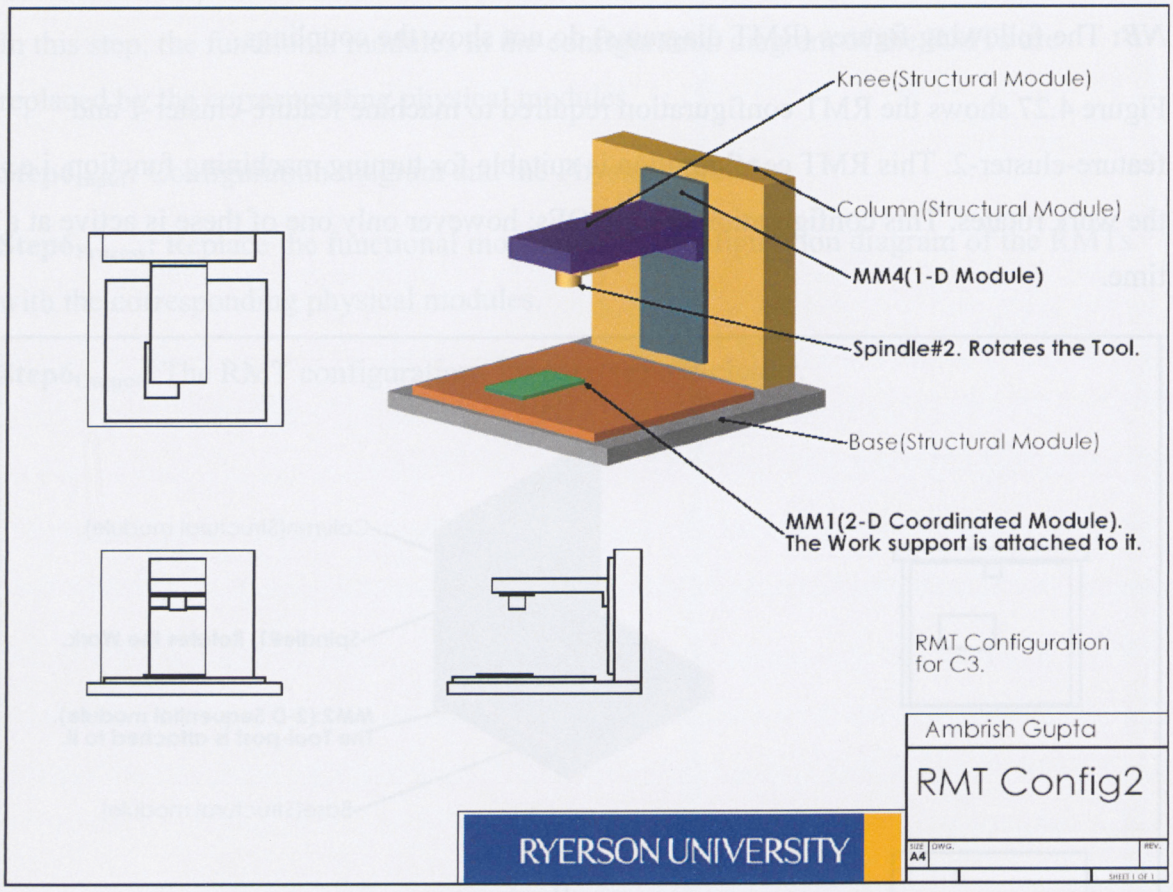


Figure 4.28: RMT Configuration for machining feature-cluster 3

Figure 4.29 presents the RMT configuration required to machine feature-cluster 4. This RMT configuration has 3-DOFs; however, only one of these DOFs is active at a time. The RMT configuration requires the tool to rotate, thus it performs the milling machining function. This RMT configuration is different from that presented in Figure 4.28 in that this one uses a smaller range 1-DOF module.

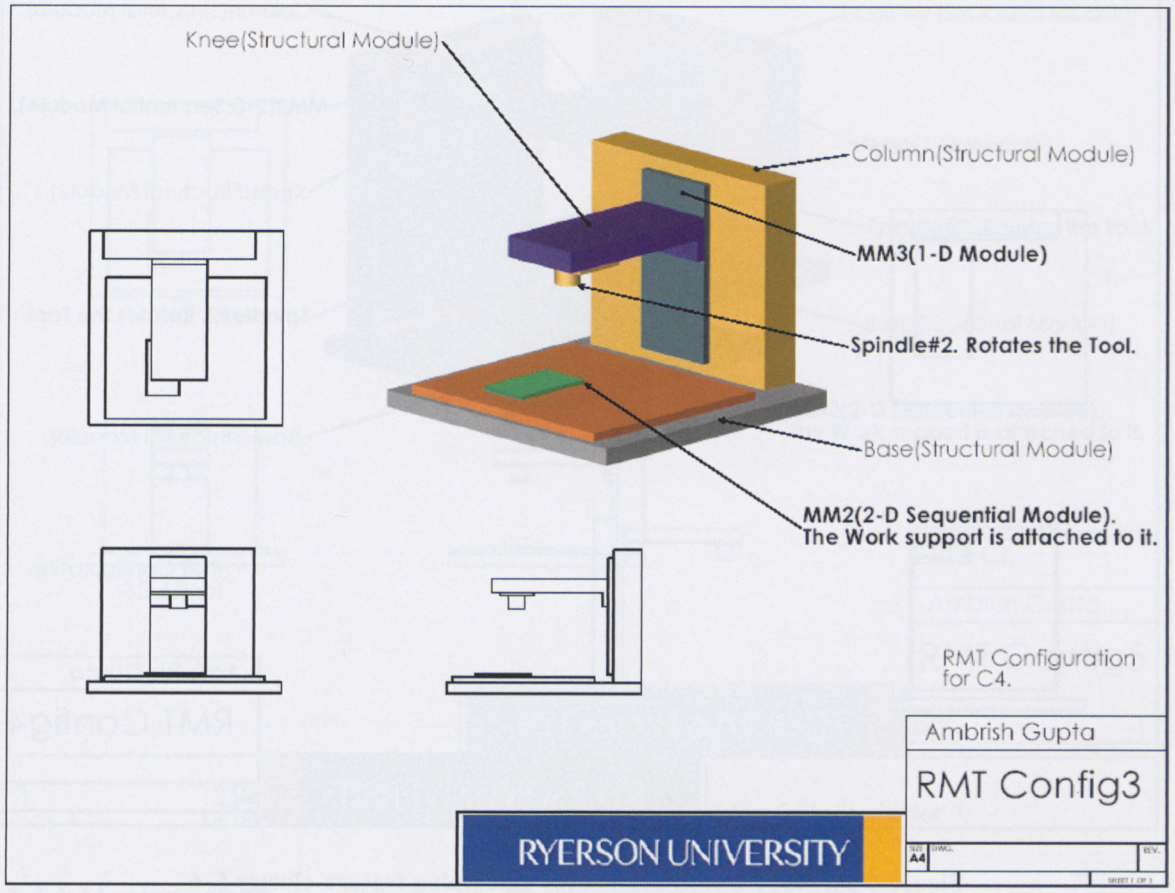


Figure 4.29-RMT Configuration for machining feature-cluster 4

Figure 4.30 presents the RMT configuration required to machine feature-cluster 5, and feature-cluster 6. This RMT configuration has 2-DOFs; however, only one of these DOFs are active at a time. The RMT configuration requires the tool to rotate, thus it performs the milling machining function.

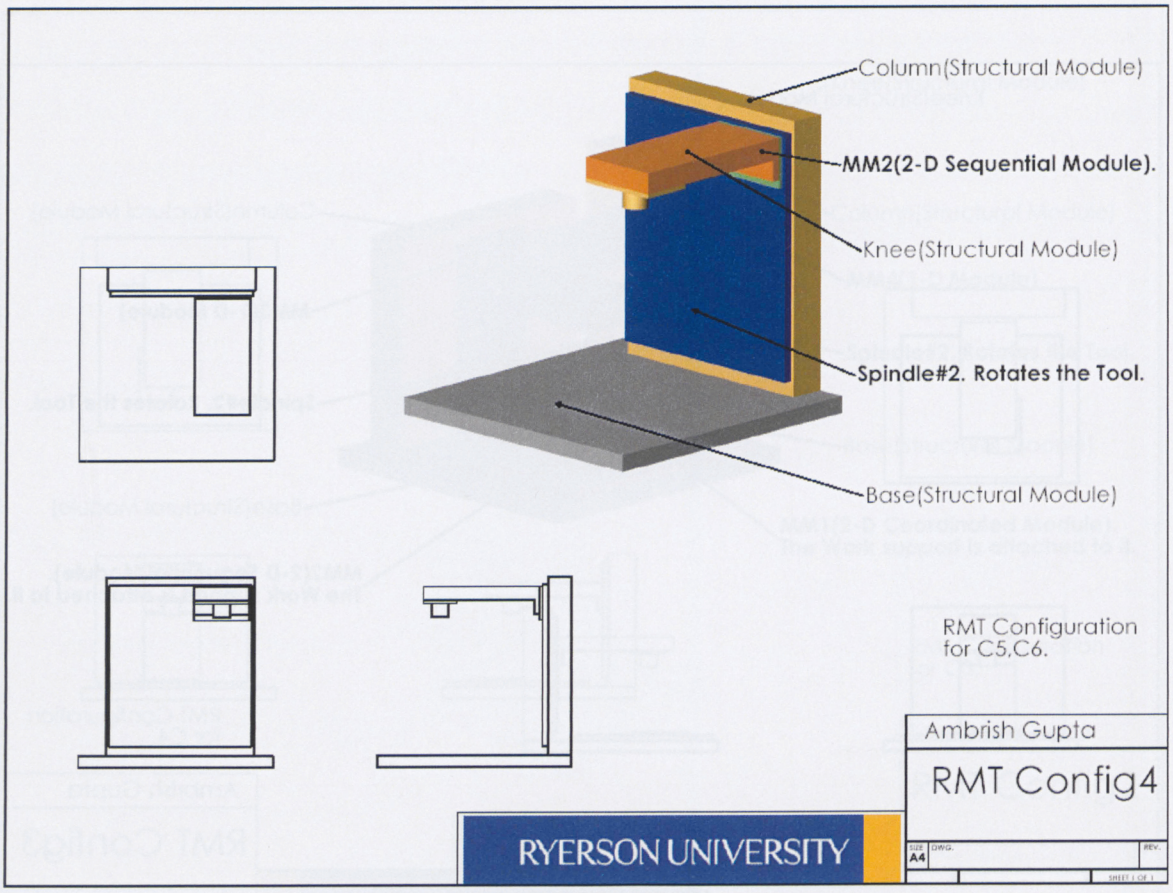


Figure 4-30: RMT Configuration for machining feature-cluster 5, 6

Figure 4.31 presents the RMT configuration required to machine feature-cluster 7. This RMT configuration has 3-DOFs; however, only one of these DOFs is active at a time. The RMT configuration requires the tool to rotate, thus it performs the milling machining function.

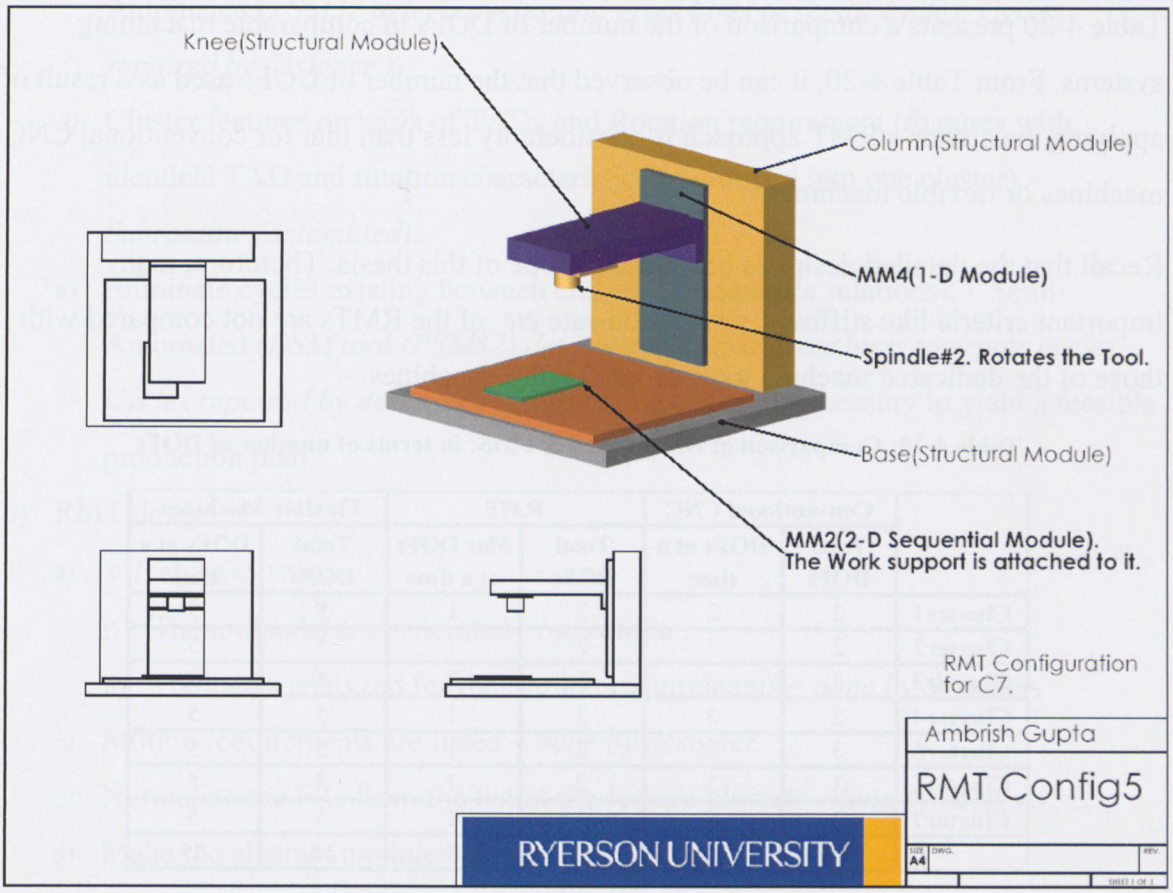


Figure 4.31: RMT Configuration for machining feature-cluster 7

4.2.4.11. Discussion

Expression 4.3 includes the sub-goals of maximizing utilization and minimizing reconfiguration effort. In addition to these sub-goals, the design objective could be extended to consider the following sub-goals:

- Minimize the weight of the modules (this calls for the mechanism to be designed elegantly so that the actuator sizes are minimum).
- Maximize the strength of the machine configurations. This indirectly means that the coupling is designed to get the maximum strength for the machine configuration.

Recall that this thesis considers only motion FRs and interfacing FRs to constitute the functional description of the motion modules. The method developed in this thesis can be extended to consider other independent FRs also, without requiring any change to the method.

Table 4-20 presents a comparison of the number of DOFs in comparable machining systems. From Table 4-20, it can be observed that the number of DOFs used as a result of applying the author's RMT approach is considerably less than that for conventional CNC machines or flexible machines.

Recall that the detailed design is beyond the scope of this thesis. Therefore, many important criteria like stiffness, production-rate *etc.* of the RMTs are not compared with those of the dedicated machine tools or the flexible machines.

Table 4-20: Comparison of DMS, RMS & FMS: in terms of number of DOFs

	Conventional CNC		RMT		Flexible Machines	
	Total DOFs	DOFs at a time	Total DOFs	Max DOFs at a time	Total DOFs	DOFs at a time
Cluster1	2	2	2	1	5	5
Cluster2	2	2	2	1	5	5
Cluster3	3	3	3	2	5	5
Cluster4	3	3	3	1	5	5
Cluster5	3	3	2	1	5	5
Cluster6	3	3	2	1	5	5
Cluster7	3	3	3	1	5	5
	19	**19**	**17**	**8**	**35**	**35**

4.2.4.12. Summary

The method for designing RMTs for machining a single part is summarized as follows:

1) Part input (as features)
2) Feature-clustering (Setup planning):
   a) ∀ mfg. feature:
      i) Identify TAD(s) – *done by designer*
      ii) Identify rotation requirement (tool/work rotates) - *done by designer.*
      iii) Determine the minimal set of TADs required for machining the part in full. – *Subroutine (automated)*

b) ∀mfg. feature pair: identify machining precedence relation and assign it a metric (The precedence relation between features is quantified: 0 = critical, 1 = for quality, 2 = for optimality.) - *done by designer*

c) Eliminate cycles existing between features in the precedence graph– Semi-Automated (*PSM32 detects cycles and gives hints to repair cycles. Cycles repaired by designer.)*

d) Cluster features on basis of TADs and Rotation requirement (features with identical TAD and rotation characteristic are grouped into one cluster). – *Subroutine (automated).*

e) Eliminate cycles existing between clusters (precedence relations). – Semi-Automated (*DSM tool (PSM32) detects cycles and gives hints to repair cycles. Cycles repaired by designer.)* . Eliminating cycles is necessary to yield a feasible production plan.

3) RMT design:

a) ∀ feature-cluster:

i) The toolpaths are generated - *Subroutine.*

ii) Toolpaths analyzed for the Motion requirements – *done by designer.*

b) Motion requirements are listed – *done by designer.*

c) Normalize the FRs from the list of the feature-clusters– *done by designer.*

d) Make the abstract modules:

i) The Normalized FRs are mapped back to the feature-clusters. (to see what FRs are employed by what feature-cluster) – *done by designer.*

ii) The Normalized FRs are clustered into meta-modules.

e) Choose a suitable configuration- *Done by Designer.*

f) Based on the motion, and interfacing requirements assign physical detail to the functional modules. The physical design should be such that the modules suit all the configurations they take involving them. - *Done by Designer.*

g) The physical modules are assembled into RMT Configurations (chosen in step 'e'). These completely designed modules are assembled back in the configurations chosen for each feature-cluster. - *Done by Designer.*

4.3. The Method for designing RMTs for machining a Part-family

In the previous section, the author developed a method to determine the functional specification of the RMTs required to machine a single part. This method was extrapolated to design RMTs for a part-family, and is presented in this section.

4.3.1. Approach

Each of the parts of the part-family can be processed according to the method developed for a single part. Then the processes can be merged once the information is free from the context of individual parts. This would be the most logical extension of the method for a single part to suit the problem of designing RMTs for a part-family.

4.3.2. The Method

The flowchart for the method (for a part-family) is presented in Figure 4.32 followed by details of the method.

Figure 4.32 presents the extrapolation of the method developed in Chapter 4 to design RMTs for a part-family. The input to the method are the parts (Part1.x, where x=a, b, c, d), which are the members of the part-family.

As can be seen, the method for a part-family is very similar to that for a single part. Each part is processed by an individual process-thread [91], which implements the method for a single part. These threads merge after the FRs for all the feature-clusters for all the parts have been determined, i.e., each member of the part-family undergoes the process (as that for a single part), and after the motion-FRs for each of its feature-clusters have been determined, these individual processes merge.

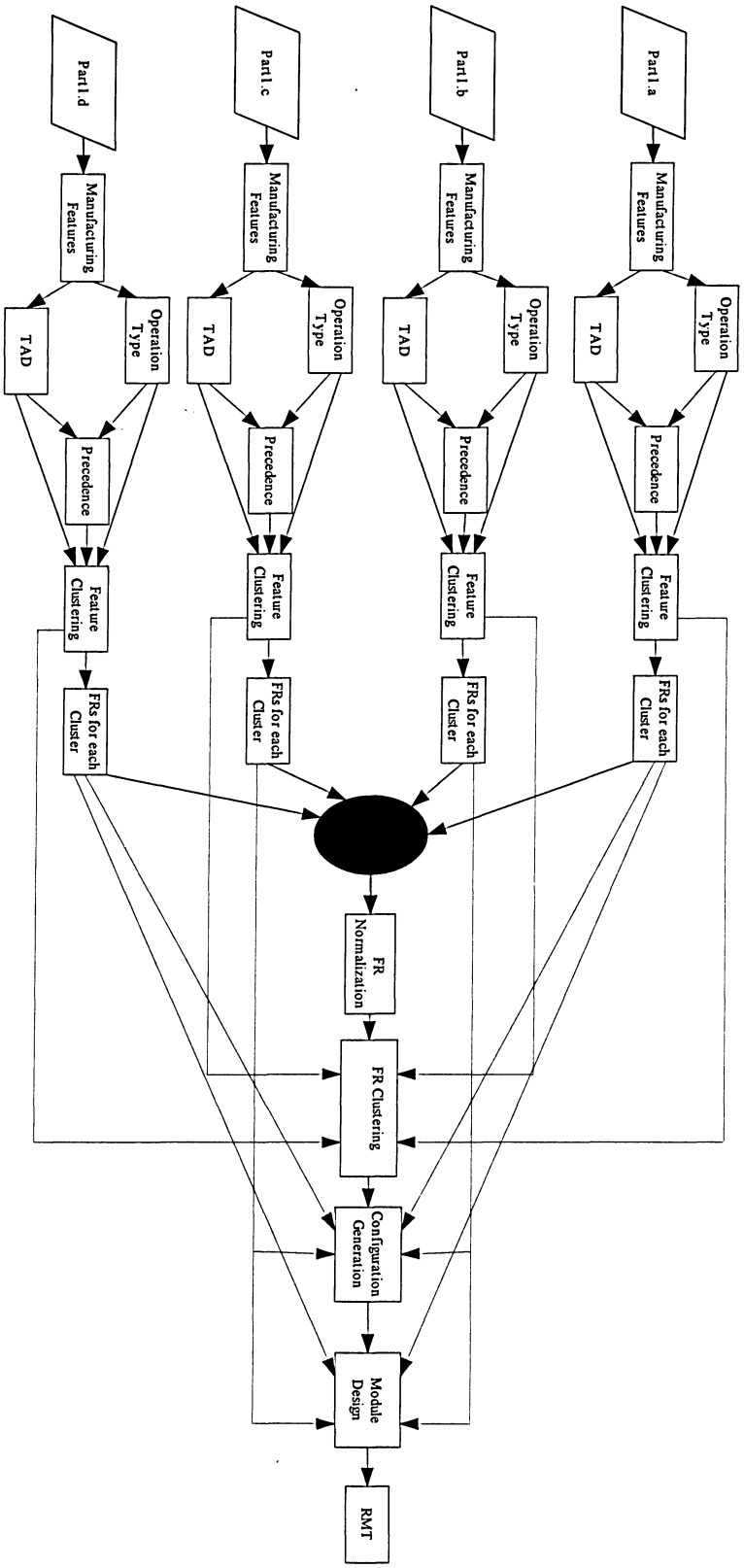


Figure 4.32 : Flowchart for designing RMTs for a part-family

4.3.3. Discussion

Recall that each feature-cluster corresponds to a RMT configuration. Each of these feature-clusters translates into motion requirements (referred to as motion FRs). These sets of motions FRs (from each part), are merged to form a hyper-set. As shown in Figure 4.32, this hyper-set is fed into the subsequent steps of normalization, clustering, and module and configuration design. Once this hyper-set has been formed, the subsequent part of the method cannot differentiate whether the set of feature-clusters came from a single part or two parts or a hundred parts. Therefore, the case for part-family degenerates to that for a single part. This implies that the example presented in Chapter 4, for a single part, is equivalent for that of a part-family. Each feature-cluster resulting from the part considered in the example is analogous to an individual part, with the previous stage of the part, as the corresponding raw-stock. For example, raw-Stock→C_1→C_2→C_3→C_4 (where each C_i is a feature-cluster), each of the C_is are equivalent to an individual part, with the part at the previous stage (with all the previous feature-clusters machined), as the corresponding raw-stock.

Common sense suggests that the parts could be merged to form a single meta-part, and RMTs could be designed for this meta-part (using the method for designing RMTs for a single part). However, merging the parts at this stage would be incorrect because the precedence constraints are defined strictly in context of the part. Merging the parts at any stage before the information is free from the part's context would lead to an undesired loss of some precedence constraints. Therefore, the processes (executing the 'method for designing RMTs for a single part', on each part of the part-family) had to be merged after the processes carry information, which was free from the part's context. Since, after the feature-clustering step, the information is free of the part's context, the processes can be merged here. However, to reduce the computational complexity of the method (which varies as the square of the number of variables/parameters/operations [58]), this thesis contended to merge the process threads after the FRs for each feature-cluster have been identified.

If a part, which has not been considered while designing RMTs, is added into the production portfolio, the whole process needs to be redone with that part included. In

other words, the new part would have to be merged into the part-family for which, the RMTs are to be designed, and the whole process redone.

4.3.4. Summary

In this section, the method developed for a single part (section 4.2), was extrapolated to suit the original problem statement. The individual members of the part-family were operated upon by the method for designing RMTs for a single part. After the FRs for the feature-clusters have been determined, these processes are merged to finally yield the RMT configurations required to machine the part-family.

5. Conclusions

5.1. Summary of Assumptions

The intent of this thesis is to develop a method for mapping a part-family onto a set of RMTs. This is a very complicated task. To simplify it, the author has made a few assumptions. These assumptions are listed below.

1) The method developed in this thesis is based on the manufacturing features. Thus, the part's geometry model needs to be converted into a feature model. This step is called feature-recognition. In the feature-recognition step, the part's model is decomposed into manufacturing features. Typically, a software agent is used to perform this task. However, due to the complexity of this task, even the most recent commercial feature recognition software like FeatureWorks® requires some human assistance. Therefore, the author assumes that the part is modeled as a 'feature model'. This simplifies the overall problem, and allows the author to focus on the main problem instead of the complexities of feature recognition. Moreover, modeling the part as a 'feature model' is quite practical because most contemporary CAD software models parts with features (i.e., 'design-with-features'). Modeling of parts with features is also compliant with STEP-AP-224. For cases where a part model is not available, the physical model can be scanned to yield a virtual model. This virtual model can then undergo the process of human assisted feature recognition. Once the features of the parts have been identified, they can be used by the method developed in this thesis.

2) This thesis uses the top-down design strategy. This strategy ensures that the system is designed specifically for the given task-set. However, this strategy is not applicable for a system with coupled components because the method would not be able to identify what functionality is manifested by which component. Therefore, the author assumes that the modules are functionally decoupled.

3) The designed RMTs may affect the choice of the machining operations or TADs for the features. Similarly, the clustering of the features may affect the choice of the machining operations or TADs for the features. This can cause iteration. Although

iterating complicates the process, it yields better RMTs. For the sake of simplicity, iterations are ignored.

4) In the feature-clustering step, manufacturing precedence is an important criterion. The author quantified the precedence relations based on their criticality. The quantification scheme used in this thesis could be replaced by any other scheme, without affecting the validity of the method developed. However, the scheme does affect the method's efficiency. The author did not consider efficiency in this work.

5) Although, a 'coordinated' module can functionally subsume a 'sequential' module, in this thesis, 'sequential' and 'coordinated' modules are considered separately. A detailed rationale for making this assumption is presented in Appendix I.

6) It is assumed that the 'material-handling system' places the part into the machine such that it is aligned with the RMT's home position. This assumption simplifies the determination of the required motions.

7) In principle, the designs of the structural modules and the material–handling system are coupled with those of the motion modules. Therefore, the task of designing the motion modules is very complex. To avoid this complication, the designs of the structural modules and the material–handling system are assumed to be decoupled from those of the motion modules. This implies that the motion modules be designed iteratively until they converge. Once the designs converge, the structural modules and the material–handling system are designed based on the converged designs of the motion modules.

8) The tool path depends on the choice of the bounding box of the part (for details, refer to section 4.3). The bounding box of the part changes as the part undergoes machining. Therefore, the bounding box needs to be updated after every machining operation. To simplify this, the bounding box of the raw-stock is assumed constant throughout all machining operations.

9) Li *et al* [44] concluded that "Flexible Kinematic Coupling" (FKC) was the best method to connect modules. Although no detailed design was performed in this thesis

(because it is beyond the scope of this thesis), it was assumed that the modules were connected by FKCs or something similar.

Relaxing these assumptions would add significant complexity to the overall problem, but render it more realistic. This thesis focused on developing a feasible method. Relaxing these assumptions is future work.

5.2. Contributions

This thesis developed a novel method to design a set of RMTs for a part-family. The method was developed by firstly scaling-down the problem, next, developing a method for the scaled-down problem, and finally extrapolating this solution to suit the original problem of designing RMTs for a part-family.

The method developed is STEP compliant throughout. For example, the parts are represented as manufacturing features; STEP-AP-224 also represents part's design data using manufacturing features. As another example, the final modules are represented as classes (in context of Object Oriented Programming); STEP-AP-233 also represents (sub)-systems as classes. Even though the detailed representation of entities does not follow any specific STEP-AP, at a conceptual level, the method is STEP compliant. As a secondary benefit, using a STEP compliant method allows it to piggyback on established STEP based computer applications. Moreover, this method can be unified with other STEP applications, and information system applications to develop an autonomous E-RMS. Such a machining system would be a reconfigurable E-Factory. That is, it would take orders from the customers via the WWW, and autonomously reconfigure, depending on the customer's demands. This grants even more feasibility to the method.

In this thesis, the author gave a novel framework of black–boxes, each containing a component of the overall design process for RMTs. A reasonable method was chosen for each of the black boxes. However, there was no pre-existing method found for clustering features, which had been developed with an intent to design the required machines. Rather, all of the surveyed feature-clustering methods assumed the availability of a pre-designed machine. Therefore, the author developed a novel feature-clustering method with an objective of designing a machining resource to machine these feature-clusters.

Moreover, DSM is used in this method. None of the surveyed works has ever used the DSM for this task.

As mentioned in the section 1.3, this thesis is limited to the functional design of the modules and the RMTs. A comprehensive functional design would require the method to consider all the functions like motions, motion-rates, manipulability, power, tribology, convertibility *etc.* The list is very long. In the timeframe of this thesis, considering all these functional requirements was not possible. Thus, the author's method considered 'motion' requirements only. However, the method developed in this thesis, scales up to include as many independent functions as the designer wishes to include.

5.3. Future Work

Optimize: In the previous section, it has been asserted that reasonable methods were used for the different black boxes, with no commitment to optimality. Thus, determining a set of optimal methods (for the black boxes to implement) is very important future work.

The author developed a novel design objective for designing RMTs. This abstract objective lacks mathematical details. Working out the fine details of this objective to result in a mathematical objective function is complicated. This would potentially require numerous simulation analyses. Once expression 4.3 is given mathematical detail, a variety of discrete optimization algorithms could be used to automatically design the modules and the RMTs. Constructing a mathematical objective function and deciding the optimization algorithm for automatically designing the modules and the RMTs is future work. Such an objective function also has a potential use in developing a resource planning/scheduling algorithm. As another secondary benefit, it could also control the level of granularity of the RMTs.

Automate: The most important future development for this method would be to automate it. In the method developed, there are numerous steps, which require human assistance, e.g., clustering individual motions to result in Atomic Modules. Therefore, automating the method would require a software agent to replace at least some of the human involvement. Moreover, since the method offers STEP compliance, developing a

software using appropriate STEP-APs would add even more feasibility to the method developed in this thesis.

Design in detail: The scope of this thesis is limited to the functional design of the motion modules and the RMTs. The most important future work is to extend this work to consider all the pertinent functional requirements and to perform the detailed design. The detailed physical designs can then be analyzed and compared with others.

Incremental design: If a new part is added to a set of parts for which the RMTs have been designed, the method needs to be re-executed with the updated set of parts. This is similar to most of the research surveyed in the field of setup planning. If a new feature is added to the part, for which setups have already been determined, the whole process has to be re-executed with the updated set of features , i.e., there is no technique for the incremental planning. An item for future work could be to do incremental planning by using a hybrid design strategy combining both 'bottom-up' and 'top-down' design strategies. In such a strategy, the existing modules are first assembled according to the bottom-up strategy to generate RMT configurations to best fulfill the tasks posed by the new part. Next, the remaining functionality is added by using the 'top-down' strategy, to customize the RMTs completely for the new part.

Miscellaneous:

The configuration diagram used in this thesis (section 4.2.4.7.) is a weak representation, mainly because it only shows the functional modules assigned to the two arms ('base to tool' and 'base to workpiece') of the modules. Although it shows the functional structure of the machine, it bears no direct congruence with the physical model of the machine, i.e., it does not show the exact arrangement/sequence of modules, the interfacing, the structural modules attached between two motion modules, *etc.* The author proposes to extend this representation such that it is represents the RMT configuration more realistically.

In section 4.2.4.8, the author asserted the shortcomings of a 2-D DSM for the task. Although DSMs of higher dimensions have been proposed in past articles, no research has been done on them. Thus, the author proposes the development of methods and operations for 3-D DSMs.

Li *et al* [44] asserted that 'Flexible Kinematic Couplings' was the best candidate for connecting modules. In a research contemporary to Li *et al*'s study, Culpepper [12] developed 'Quasi Kinematic Coupling' (QKC). The author anticipates that QKC would give a performance superior to that of FKC. A detailed investigation would be required to determine the validity of this anticipation. Therefore, it is proposed to compare FKC and QKC in regard to the coupling of modules in a RMT.

Having a reconfigurable controller for a RMT is an enticing concept. FPGA based controllers would open the possibility of evolving automatically and calibrate themselves. They can also be coupled with an inspection agent for them to adjust their parameters continually (in order to compensate for errors). A detailed investigation of their potential use in RMTs is future work.

The method developed in this thesis does not apply to sculpted features. This is a serious drawback. The author proposes to extend the method such that it is applicable to sculpted features also.

Zhong [111] presented a method for comparing different machining systems. As a future work, the author proposes to compare the RMTs resulting from this method with other contemporary machining systems and quantify the economic advantage.

Appendix-I

This appendix describes the sequential and coordinated modules, with an intent to present the rationale for modeling them separately. Finally, a summary is presented.

I. Coordinated Motion Module

In this section, the author presents the definition, the attributes of this category of modules and finally the internal schematics of a 2-D coordinated module.

I.i. Definition: For this category of modules, all the joints embedded in the module actuate simultaneously to manifest the designated task-set.

I.ii. Attributes:

1) It has a motor for each actuated joint. If a uni-drive mechanism [33] is used, this category of modules requires simultaneous transfer of mechanical power to all the joints.

2) This category of modules preferably uses non-cartesian mechanisms, because all the joints are required to actuate simultaneously.

3) A non-cartesian mechanism typically yields higher velocity-amplification-ratios. This implies that the actuators can be of lesser capacity as compared to those in a Cartesian mechanism. However, using such mechanisms has a drawback of complicating the actuator-design process, because the motor designs are coupled.

4) The controller used in this category of motion modules is complicated, because it has to control all the joints simultaneously. Moreover, since a non-cartesian mechanism is typically used, the control strategy is more intricate. Since the control system needs to handle different actuators simultaneously, the size of the communication and control buses [96] and the clock-frequency of the controller are also critical to the module's design.

I.iii. Internal Schematic:

Figure A shows the internals schematics of the 2-D coordinated module.

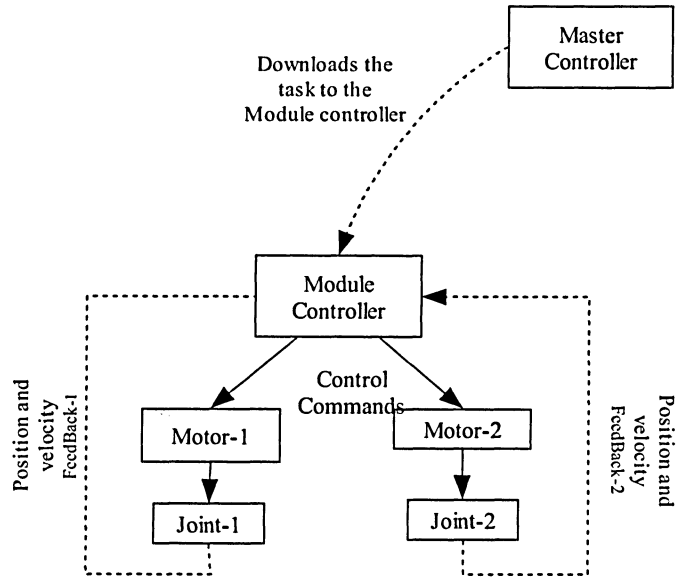


Figure A: 2-D Coordinated Module

II. Sequential Motion Module

In this section, the author presents the definition, the attributes of this category of modules and finally the internal schematics of a 2-D sequential module.

II.i. Definition: For this category of modules, only one of the joints actuates at a time to fulfill a subset of the designated task.

II.ii. Attributes:

1) It has a single motor. The mechanical power from the motor is routed via a multiplexer/channel-reconfigurator to one of the joints. In the case of a uni-drive mechanism, these modules demand a sequential power transfer to the joints.

2) These modules typically use a cartesian mechanism. The main reason being that, for cartesian mechanisms, each of the joints corresponds to a single operation space DOF.

3) Since the jacobian matrix for a cartesian mechanism is diagonal, the motor designs are not coupled. This further implies that determining one motor's capacities is isolated from that of others.

4) The controller needs to control only a single joint at a time. However, it also has to control the multiplexer/channel-reconfigurator, which routes the flow of mechanical

power from the motor to one of the joints. Therefore, the function requirements of the controller are different from those a coordinated module.

II.iii. Internal Schematic:

Figure B shows the diagram for the internals of the 2-D sequential module. The module's controller commands the motor, and the multiplexer/channel-multiplexer.

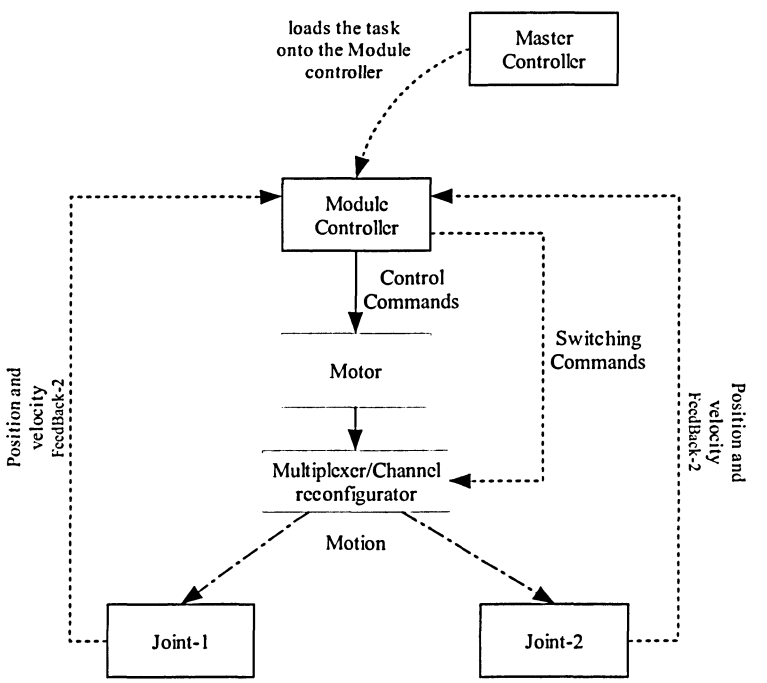


Figure B: 2-D Sequential Module

6.3. Summary

As demonstrated in this appendix, there is a considerable difference between the coordinated and sequential modules. Therefore, the author chose not to subsume the sequential module with the coordinated module. In this thesis, the method resulted in 2-D sequential and 2-D coordinated modules.

Glossary

This chapter presents the definitions of the terms used in the thesis.

Definitions:

Abstract Class (in context of Object-Oriented programming): - Abstract-classes are used to represent abstract concepts or entities. "The incomplete features of the abstract class are then shared by a group of sibling sub-classes which add different variations of the missing pieces. Abstract classes are superclasses which contain abstract methods and are defined such that subclasses are to extend them by implementing the methods. The behaviors defined by such a class are *'generic'* and much of the class will be undefined and unimplemented. Before a class derived from an abstract class can be instantiated, it must implement particular methods for all the abstract methods of its parent classes." [88]

***NB*:** In this thesis, the 'module' class is modeled as the abstract-class. All the specific types of modules derive from this 'generic' class. These module types add specific functions to the general fields and operations of the 'module' abstract-class.

Agent: - An agent is an autonomous system, which could be a computer program, or a robot, or a human. It is designed/trained to perform a task without any external assistance. An agent is typically sensitive to its operational environment. The agents are functionally independent, and in context to a multi-agent system, they can collaborate with other agents to produce complex system behaviors.

Ashby's theory of variation: "A system-model or a control-system can model or control something to the extent that it has sufficient internal variety to represent." [65]. It is analogous to adding knowledge a model/controller. If the model/control-system uses fuzzy logic, adding variation is analogous to adding new rules or adding more membership functions. Please note that adding more membership functions would imply addition of more rules. If the model/control-system implements a neural-network, variation can be added by appending neurons. These neurons can be added into pre-existing layers or may form an altogether new layer of neurons.

Bounding Box: - It is a 3-dimensional-orthogonal figure, which completely encloses the 'convex-hull'.

BUS: - "One of the sets of conductors (wires, PCB tracks, or connections in an integrated circuit) connecting the various functional units in an electronic system. There are buses both within the CPU and with connecting it to external memory and peripheral devices. The data bus, address bus and control signals, despite their names, really constitute a single bus since each is useless without the others." [96]

***NB*:** This thesis considers sequential and coordinated modules separately. As mentioned in 'Appendix I', the controller of a coordinated motion-module is more complex. The functional requirements of the control-system pose some requirements on the bus-width. This is an important parameter while designing the control-system for the coordinated modules.

Case Based Reasoning: "It is a technique which looks for previous examples which are similar to the current problem. This is useful where heuristic knowledge is not available. This technique analyzes the knowledge available in the form of examples to find a solution to the problem. A few of the key research areas are efficient indexing, how to define 'similarity' between cases, and how to use temporal information." [97]

Cellular Automata(CA): - "Cellular automata are mathematical models which represent complex natural systems containing large numbers of simple identical components, with local interactions. They typically have 'finite state machines' (FSMs) as their components, i.e., CA is equivalent to a lattice of FSMs. The states of these FSMs are synchronously updated according to certain rules. These rules map the previous state of a particular FSM, and its neighboring ones, to its current state." [92]

Chain reconfigurable robots: - "These robots make themselves by attaching and detaching chains of modules to and from themselves. Each chain is always attached to the rest of the modules at one or more points. Nothing ever moves off on its own. The chains may be used as arms for manipulating objects, legs for locomoting, or short tentacles for both manipulation and locomotion. E.g., PolyBot. A chain robot has already demonstrated locomotion by rolling like a tank tread, climbing stairs, slithering like a snake, climbing like a caterpillar, and walking like a spider." [63]

Class: - (in context to Object-Oriented programming) "A class is a template that describes the underlying structure of a group of objects. A class specifies the data items each object of the class contains and the operations or methods that can be performed on each object belonging to the class." [89]

CLIPS: - This is an acronym for "**C** Language **I**ntegrated **P**roduction **S**ystem." NASA developed this software. It provides a complete environment for developing expert systems. The knowledge base is fed into the expert system in the form of predicate logic, if-then rules, templates, and objects. "CLIPS's inference-engine implements the standard forward-chaining pattern-matching algorithm." [59]

CLUTO: - "CLUTO is a software package for clustering data. It provides three different classes of clustering algorithms that operate either directly in the object's feature space or in the object's similarity space. These algorithms are based on the **partitional**, **agglomerative**, and **graph-partitioning** approaches." [34]

Cover– (in the context of Boolean algebra) "Cover of a function is defined as a collection of implicants that account for the valuations for which a given function is equals 1. More than one cover may exist for a function. The set of minterms (of a boolean function) is a cover (of the boolean function)" [21, 28].

Espresso: - A 2-staged heuristic boolean minimization algorithm. "In practice, espresso finds an answer very close to the minimum using dramatically shorter execution times than exact techniques such as Quine-McCluskey." Espresso offers the following advantages over 'Quine-Mc-Cluskey algorithm.

1) It guarantees to solve for the minimum number of product terms and heuristically minimize the number of literals.
2) "It does not require an exponential amount of computational time and memory as a function of number of inputs as required by Mc-Cluskey algorithm." [21]

FPGA: -"An FPGA consists of an array of logic elements, either gates or lookup table RAMs, flip-flops and programmable interconnect wiring. Alternatively, an FPGA is defined as a generic array of logic gates (transistors), instantly "wired" to model an

'application personality', i.e., a chip with a fixed array of logic gates connected by programmable soft links; the links are programmed for the FPGA to execute a particular behavior." [98]

GA (Genetic Algorithm): - "A discrete optimization technique with operators inspired by organic evolution. GA works by creation of an electronic organism as a binary string ("chromosome") and then using genetic and evolutionary principles of fitness-proportionate selection for reproduction (including random crossover and mutation). This gives it the capability to search enormous solution spaces efficiently. " [90] Although they seem robust to the choice of the parameters and the intitial population set, yet they have the potential to be decieved and converge at sup-opitmal points. For more details on the 'Minimally Deceptive Problem'(MDP) and other limitations of GAs, the reader can refer to [20].

Implicant: - "A product term that indicates the input valuation, for which a given boolean function equals 1. Minterms are implicants" [21].

Jacobian matrix: - The mapping of the joint velocities to end effector's velocities. It is a function of the joint variables; thus, it depends on the instantaneous joint values.

Joint (Actuator) Space: - an 'N' dimensional space (where 'N'= number of actuators in the mechanism) constituted of the joint/actuator variables.

Lattice Robots: - "Robots that can change shape by moving into positions on a virtual grid or lattice. All the modules remain attached to the robot, at all times. Planning and control issues become less complex because the modules can move only to neighboring positions within a lattice instead of to any arbitrary position".[63]

Literal: - "A *literal* is either a boolean variable or its negation" [21].

Manipulability (also called 'Dexterity', 'Velocity Amplification Factor'): - It is defined as the measure of the volume of the Velocity ellipsoid for a particular set of joint values. Points/regions in the workspace corresponding to a 'manipulability' being 'zero' correspond to singularities.

Manipulability is a critical criterion for mechanism design. It affects the mechanism's topology and the actuator's size.

In this thesis, this criterion is not considered because the detailed design is beyond the scope.

Manufacturing Feature: - Volume of material to be removed by a machining operation. Every feature has a corresponding set of possible machining operations. The mapping of feature onto the set of operations depends on various factors and parameters like the part's material, feature's geometry, and tolerance relations with other features.

Manufacturing operation: - An operation involving active motions of the machine tool and the fixtured work-piece. The relative motions between the tool and the work-piece, machine the feature. The operation also contains information about the rate and ranges for active motions *etc.*

Maxterm:- "A *maxterm* of **k** variables is a disjunction (Boolean OR) of **k** literals, where each variable shows up exactly once" [21].

Min-Cover:- "The cover consisting of the smallest subset of prime implicants" [28].

Min-Cut algorithm: - This is a graph-partitioning algorithm. It implements a heuristic to minimize the cost of partitioning a graph into a specified number of sub-graphs. The cost of partitioning is a function of the number of links cut and the difference in the sizes of the sub-graphs [76].

Minterm:- "A *minterm* of **k** variables is a conjunction (Boolean AND) of **k** literals, where each variable shows up exactly once"[28].

Module: - This is defined as a functionally independent, electro-mechanical assembly. The modules are assembled to yield RMTs. The modules cooperate with others to manifest the required tasks. In this thesis, the modules are only considered in context of the motion they provide. Therefore, in this thesis, they are referred to as **Motion Modules**.

Object: - (in context to Object-Oriented programming) "An instance of a class is an object. It could be said that a class is a blueprint, and an object is a house. As another example, if humanity were a class, then {the author} would be an instance of the class {humanity}" [89].

Operation Space (OS): -The space of the end-effector (of a mechanism). It can be a maximum 6-Dimesional.

Part-family: - This is defined as a set of parts having similar manufacturing features. It is also defined as parts with a specified predicate of variation [55].

Prime Implicant: - "An implicant (a minterm) is called 'prime' if it cannot be combined into another implicant that has fewer literals. It is impossible to delete any literal in a prime implicant and still have a valid implicant" [21]. A boolean function can be reduced to a sum of product terms called prime implicants of the function.

Principal component analysis (PCA): - "This involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components*. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible."[31]

Material handling system/agent: - It performs the activities of (un)fixturing the part into the RMT configuration. The fixturing aspect of this system could also be reconfigurable.

Quine Mc-Cluskey's minimization method:- A 2-stepped algorithm based, boolean minimization method. It is used when the number of variables is high, because using Karnaugh maps would be infeasible. For more details, refer to [21].

Self-Organizing Map (SOM): - "The SOM is a special neural network that transforms a model of arbitrary dimensionality into the responses of one or two-dimensional (2-D) arrays of neurons, and to execute the result in a topological order." [39]

Space Elevator: - This is a system for transporting a payload from a point (in the sky) to another (in outer space). It includes a first structure, located at a first relatively fixed, non-zero orbital distance from the surface of the earth, for receiving payloads. Its second structure is located at a second relatively fixed orbital distance from the surface of the earth and receives payloads, where the second distance is greater than the first distance. The third structure located near the center of gravity of the combined apparatus provides a platform for storing and/or processing payloads. A payload transporting apparatus is

disposed between and interconnecting the first and third and second and third structural means. [2]

Statically determinate structure: - A statically determinate structure is defined as a structure for which the internal forces and reactions can be determined by considering nothing more than equations of equilibrium.

STEP: - STEP is an acronym for **ST**andard for the **E**xchange of **P**roduct data. Its official name is ISO-10303. It provides a platform independent representation of product data throughout its life cycle. This representation is suitable for file exchange.[95]

Moreover, it serves as a basis for implementing product databases and for archiving data. "This standard is implemented within computer software associated with particular engineering applications and so its use and function will be transparent to a designer. ISO 10303 descriptions are information models that capture the semantics of an industrial requirement and provide standardized structures within which data values can be understood by a computer implementation. The exchange of data is one of the uses for a standardized representation, but it is not the only use. STEP Application Protocols (APs) specify the requirements for data for a specific engineering application in a standardized representation derived from the Integrated Generic Resources. Specific Application Protocols are implemented for use with relevant engineering application software" [60].

The following table presents the different APs and the objects they represent.

Table: STEP APs and brief introduction [95, 60]

AP#	Name	Description
203	Configuration controlled design	AP203 is designed for the exchange between application systems of configuration controlled 3D designs of mechanical parts and assemblies.
214	Core Data for Automotive Mechanical Design	AP214 is for mechanical design processes. It is a super set of AP203, including color and layer information
224	Mechanical product definition for process plans using machining	The AP specifies the information requirements for the representation and exchange of information needed to define product data necessary for manufacturing single piece mechanical parts
233	Systems Engineering data representation	The AP specifies the system engineering aspect and is an adaptation of UML for a generic represention of mechanical systems. The AP is completely Object-oriented.
238	STEP-NC	It specifies the machining aspect of the part in terms of machines, operations, tools etc. The main benefit of this AP is its platform independence. It is an objectified representation and carries information on WHATs and not HOWs.

Structural modules: - These are defined as the modules that add rigidity to the machine but do not provide any motions.

Subsumption: - This is defined as the superceding of a module (in this thesis specifically, in the aspect of 'motion ranges') by another single module or an assembly of other modules.

Tearing Advice: - The PSM-32 software, uses embedded heuristics to identify the potential benefit of ignoring a particular relation between a pair of (sub)systems. The relations between the (sub)systems are analogous to edges of the graph-dual of the DSM. The software ranks the different inter-relations according to the potential benefit of breaking an edge. Breaking an edge is analogous to ignoring the relation between the (sub)systems connected by the edge. The heuristics are purely mathematical and do not have engineering decision-making capability. The designer requests tearing advice from the software and makes decisions based on his engineering knowledge.

Thread : - (in the context of Computer Science) "Threads are similar to processes, in that both represent a single sequence of instructions executed in parallel with sequences, either by time slicing or multiprocessing. Threads are distinguished from traditional multi-tasking processes in that processes are typically independent, carry considerable state information, and interact only through system-provided inter-process communication mechanisms. This allows a program to be split into two or more simultaneously running tasks. Multiple threads, on the other hand, typically share the state information of a single process, share memory and other resources directly." [91] An advantage of a multi-threaded program is that it can operate faster on machines that have multiple CPUs, or across a cluster of machines or a Hyper-Threading enabled processor [29].

Unidrive mechanism: - A Unidrive mechanism has a motor that drives a flexible shaft [33]. The joint actuators draw mechanical power from this shaft via a transmission (clutch and a gear - box). In context to such a mechanism, the difference between the coordinated and sequential reduces to tapping power simultaneously or sequentially , i.e., either the clutches are engaged simultaneously or not. From the design aspect, the

'Sequential' module would require a motor of lesser capacity. The control system of the 'coordinated' module would be more complex.

Velocity ellipsoid: - This is a mapping of a hyper-sphere of joint velocities into the operation space. The ellipsoid's dimensions are given by eigenvalues of the Jacobian.

VGT (Variable Geometry Truss): - This is a reconfigurable Parallel robot. TetroBot [23], and Xi *et al*'s [100] robot belong to this category.

Web-Crawler: - "An automated software-application that follows links to visit web sites on behalf of search engines or directories. Crawlers then process and index the code and content of a web page to be stored in the search engine's database. E.g., Googlebot is the crawler that travels the web finding and indexing pages for the Google search engine." [94]

References

[1] Angeniol B, Vaubois G.C, & Texier J.L. (1988)., "Self-Organizing Feature Maps and the Traveling Salesman Problem", *Neural Networks, 1*: 289-293.

[2] Boyd R.R, & Thomas D.D (2002) , "Space Elevator", *US Patent: 6,491,258.*

[3] Bradley T.K. (2002) , "Utilization of Dependency Structure Matrix Analysis to Assess Implementation of NASA's Complex Technical Projects", *Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA.*

[4] Brown, K. N.(1997), "Grammatical Design", *IEEE Expert, Special Issue on Artificial Intelligence in Design* 12(2): 27-33.

[5] Browning, T.R.(2001), "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions", *IEEE Transactions on Engineering Management*, 48(3): 292-306.

[6] Browning, T.R.(2002), "Process Integration Using the Design Structure Matrix", *Systems Engineering*, 5(3): 180-193.

[7] Butler Z., Kotay K., Rus D., & Tomita K. (2002.), "Cellular Automata for Decentralized Control of Self-Reconfigurable Robots", *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington D.C.*: 809-816.

[8] Canadian Space Agency, "The Evolution of Canada's Robot Arms", Viewed August'2003,
<http://www.space.gc.ca/asc/eng/csa_sectors/human_pre/iss/canadarm2/evolution.asp>.

[9] Castano A., Behar A., & Will P.M. (2002), "The Conro Modules for Reconfigurable Robots", *IEEE/ASME Transactions on Mechatronics,* 7(4): 403 -409.

[10] Castano A., & Will P. (2000), "Mechanical Design of a Module For Reconfigurable Robots", *Proceedings of 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 3: 2203 -2209.

[11] Chick S.E., Olsen T., Sethuraman K., Stecke K., & White C.C. (2000), "A Descriptive Multi-Attribute Model For Reconfigurable Machine System Selection Examining Buyer-Supplier Relationships", *International Journal of Agile Management Systems 2(1)*: 33-48.

[12] Culpepper M.L. (2000), "Design and Application of Compliant Quasi-Kinematic Couplings", *Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.*

[13] Dabling J.G., & Chase K.W. (2002), "Performing Tolerance Analysis of 3D Assemblies Using ADAMS", *Mechanical Dynamics User Conference.*

[14] DeGaspari J., "All in the Family", *Mechanical Engineering Magazine* –Feb 2002, Viewed August-2003: <http://www.memagazine.org/backissues/feb02/features/allinthe/allinthe.html>.

[15] DeHon A., & Wawrzynek J. (1999) , "Reconfigurable Computing: What, Why, and Implications For Design Automation", *Proceedings of the 36th ACM/IEEE conference on Design automation :* 610 – 615.

[16] Demey, H. V., & Derache H. (1996), "Determining Setups for Mechanical Workpieces", *Robotics and Computer-Integrated Manufacturing,* 12(2): 195-205.

[17] Dong Q., & Whitney D.E. (2001), "Designing a Requirement Driven Product Development Process", *Proceedings of DETC 2001: ASME 2001 International Design Engineering Technical Conferences 13th International Conference on Design Theory and Methodology September 9-12, 2001, Pittsburgh, PA.*

[18] European STEP-NC Consortium. "STEP–NC: Official Site of STEP-NC", Viewed August' 2003, <http://www.step-nc.org/>.

[19] Fitzhorn P.(1992), "Formal Graph Languages of Shape.", *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 4 (3): 151-163.

[20] Forrest S., & Mitchell M. (1993), "What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation", *Machine Learning*, 13: 285- 319.

[21] Givone D.D.(2002), "Digital Principles and Design with CD-ROM", McGraw-Hill Higher Education, New York.

[22] Goldratt E.M.(1999),"Theory of Constraints", North River Press Publishing Corporation, Great Barrington, MA.

[23] Hamlin G.J , & Sanderson A.C.(1998), "Tetrobot : a Modular Approach to Reconfigurable Parallel Robotics", Kluwer Academic Publishers, Boston.

[24] Helo P.T. (2002), "Feature Based Configuration - Configuration Sequence Analysis with DSM,", *Proceedings of the Fourth MIT DSM International Workshop (Cambridge, MA).*

[25] Hilmola O.P.K., & Helo P.(2000), "Improving Product's Time-to-Market - An Application of DSM and TOC". *Proceedings of the Second MIT DSM International Workshop, Boston, MA.*

[26] Hilmola O.P.K, Manuksela A., & Helo P.(2003), "The Economic Nature of Feedback Loops- Some Experiments With Ashby's 'Systems Thinking' and DSM", *Proceedings of the fifth MIT DSM International Workshop, Cambridge, UK.*

[27] Huang S H. (1998), "Automated Setup Planning for Lathe Machining", *Journal of Manufacturing Systems,* 17(3): 196-208.

[28] Hugue M. (2002), "Implementing Boolean Functions", Viewed August'2003 <http://www.cs.umd.edu/class/spring2003/cmsc311/Notes/Comb/func.html>.

[29] Intel Corporation, "Hyper-Threading Technology" , Viewed- August 2003 <http://www.intel.com/technology/hyperthread/>.

[30] Jantapremjit, P., & Austin, D. (2001), "Design of a Modular Self-Reconfigurable Robot", *Australian Conference on Robotics & Automation*, Sydney: 38-43. Viewed-Jul 2003, <www.araa.asn.au/acra/acra2001/Papers/Jantapre.pdf>.

[31] Jolliffre I.T. (2002), "Principal Component Analysis", Springer Verlag, Berlin.

[32] Kapitaniak, Tomasz. (1998), "Chaos for Engineers : Theory, Applications, and Control", Springer Verlag, Berlin.

[33] Karbasi, H., Khajepour, A., & Huissoon, J.P. (2001), "Design and Simulation of a Uni-Drive Modular Robot", *MMO workshop on reconfigurable manufacturing, Hamilton, ON.*

[34] Karypis G.(2002) , "CLUTO: Clustering Package for High Dimensional Data sets", Viewed August'2003, <http://www-users.cs.umn.edu/~karypis/cluto/>.

[35] Katz, R., & Moon, Y.M.(2000), "Virtual Arch type Reconfigurable Machine Tool Design", *Technical report, University of Michigan, Ann Arbor.* Viewed- July 2003: <http://eclipse.engin.umich.edu/Publications/PubFiles/TA3/VirtualRMTReport_41.pdf>.

[36] Katz R., Yook J., & Koren Y. (2002), "Control of a Non-orthogonal Reconfigurable Machine Tool", *Technical report, University of Michigan, Ann Arbor.* Viewed- July 2003:<eclipse.engin.umich.edu/Publications/PubFiles/TA3/CCC%20paper%20as%20Sent%20to%20journal.pdf>.

[37] Kefalas P., Eleftherakis G., & Kehris E.(2001), "Modular Modeling of Large-Scale Systems using Communicating X-Machines." *8th Panhellenic Conference on*

Informatics, Cyprus. Viewed July'2003:
<http://www.city.academic.gr/material/academic_staff/computer_science/eleftherakis/html/epy8.html>

[38] Kelly T., & Littman J. (2001), "The Art of Innovation", Doubleday, New-York.

[39] Kohonen T. (2001), "Self-Organizing Maps" 3rd Extended Edition, Springer Series in Information Sciences (30), Berlin.

[40] Koren, Y., & Kota, S. (1999) ,"Reconfigurable Machine Tool", *U.S.Patent No. 5,943,750.*

[41] Koren, Y., & Ulsoy, A.G.(2002), "Vision, Principles and Impact of Reconfigurable Manufacturing Systems", *Powertrain International: 14-21.*

[42] Landers R.G., Min B.K.,(2001),"Development of a Prototype Reconfigurable Machine Tool", *CIRP 1st–International Conference on Reconfigurable Manufacturing*, Ann Arbor, Michigan, Viewed- July 2003:
<web.umr.edu/~landersr/PAPERS/CIRPRM01.pdf>.

[43] Larman C. (2001), "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)" , Prentice Hall PTR, New-York.

[44] Li H., Landers R., & Kota S.(2000),"A Review of Feasible Joining Methods for Reconfigurable Machine Tool Components, *Japan-USA Symposium on Flexible Automation*, Viewed- July 2003: <web.umr.edu/~landersr/PAPERS/JUSA00b.pdf>.

[45] Ling M. (2001), "Patterning Algorithm for Operation clustering for Reconfigurable machining systems", *PhD Thesis, University of Michigan. Ann Arbor, Michigan.*

[46] Lockledge C., & Salustri F.A. (2001), "Restructuring Design Communication Using a Design Structure Matrix", *Proceedings of the 13th International Conference on Engineering Design, : 27–34.*

[47] Lucas M.R., Endsley E.W., & Tilbury D.M. (2000), "Modular Control For Reconfigurable Machine Tools: Integrating Servo And Logic Control", *Proceedings of the Japan-USA Symposium on Flexible Automation July 2000, Ann Arbor, Michigan*, Viewed- July 2003: <www-personal.engin.umich.edu/~tilbury/papers/let00jusfa.pdf>.

[48] Massachusetts Institute of Technology(2001), "The Design Structure Matrix – DSM", Viewed- July 2003: <http://www.dsmweb.org/>.

[49] MD Robotics (2001), "Mobile Servicing System". Viewed August-2003: <www.mdrobotics.ca/pdf_files/MSS_DS.pdf>.

[50] Mehrabi M.G., Ulsoy A.G., & Koren Y. (2000), "Reconfigurable manufacturing Systems: Key to Future Manufacturing", *Journal of Intelligent Manufacturing* 11(4): 403-419.

[51] Michael J. (1996), "Fractal Robots", Viewed- July 2003: <http://fractal-robots.com/>.

[52] Ming X. G., & Mak, K. L. (2000), "Intelligent Setup Planning in Manufacturing by Neural Networks Based Approach", *Journal of Intelligent Manufacturing*11: 311-331.

[53] Mohan, S.N. (2002) "Managing Unmanned Flight Projects Using Methods in Complex Product Development", *IEEE : Aerospace Conference Proceedings,* 7: 3473 -3488.

[54] Moon S.K., Moon Y.M., Kota S., & Landers R.G. (2001), "Screw Theory Based Metrology for Design and Error Compensation of Machine Tools", *ASME 2001 Design Engineering Technical Conference, Pittsburg, PA.*

[55] Moon Y.-M. (2000), "Reconfigurable Machine Tool Design: Theory and Application", *Ph. D. Dissertation, University of Michigan, Ann Arbor, Michigan.*

[56] Moon Y.M., & Kota S. (2002), "Automated Synthesis of Mechanisms Using Dual-Vector Algebra", *Mechanism and Machine Theory* 37(2): 143-166.

[57] Moon Y.-M., & Kota S. (2001), "Reconfigurable Power Spindle", *U.S. Patent : 6,309,319.*

[58] Munson J.C. Khoshgoftaar, T.M.(1992), "Measuring Dynamic Program Complexity", *IEEE Software* 9(6): 48-55.

[59] National Aeronautics and Space Agency, "CLIPS, a Tool for Building Expert Systems", Viewed August-2003: <http://www.ghg.net/clips/CLIPS.html>.

[60] National Institute of Standards and Technology, "The STEP Project", Viewed August'2003: <http://www.nist.gov/sc4/www/stepdocs.htm>.

[61] Ong S.K., & Nee A.N.Y. (1997), "Automatic Setup Planning in Machining Operations", *Journal of materials processing technology 63: 151-156.*

[62] Pantapolous G., "Holographic Information Systems", *PhD Thesis, 2003, California Institute of Technology, Pasadena.*

[63] PARC (1997), "Modular Reconfigurable Robotics", Viewed- July 2003: <http://www2.parc.com/spl/projects/modrobots/>.

[64] Poon J. and Maher M.L. (1996), "Emergent Behaviour in Co-Evolutionary Design", *Artificial Intelligence in Design,* : 703-722.

[65] Principia Cybernetica Web (2001), "The Law of Requisite Variety", Viewed July - 2003: <http://pespmc1.vub.ac.be/REQVAR.html>.

[66] Rogers J. L. (1997), "Reducing Design Cycle Time and Cost through Process Resequencing," *11th International Conference on Engineering Design,* 1: 193-198.

[67] Roush W. (2003), "Computers That Speak Your Language", *Technology Review* 106(5): 32-39.

[68] Rus D., & Vona M. (2000), "A Basis for Self-Reconfiguring Robots Using Crystal Modules", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems,* 3(31): 2194 -2202.

[69] Sarma S.E., & Wright P. (1996), "Algorithm for Minimization of Setups and Tool Change in Simply Fixturable Components for Milling", *Journal of manufacturing systems* 15(2): 95-112.

[70] Saaty T.L. (1994), "Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process", RWS Publications, Pittsburgh, PA.

[71] Sela O.G.,Dombre E., & Benhabib B. (1997), "A Reconfigurable Modular Fixturing System for Thin-Walled Flexible Objects", *International Journal of Advance Manufacturing Technology*, 13(9): 611-617, Viewed- July 2003: <www.mie.utoronto.ca/labs/ciml/projects/ design/Fixturesela.pdf>.

[72] Shirinzadeh B. (1995), "A CAD-based Hierarchical Interference Detection Among Fixture Modules in a Reconfigurable Fixturing System" , *International Journal of Robotics and Computer-Integrated Manufacturing*, 12(1): 41-53.

[73] Smith R.P., & Eppinger S.D. (1997) , "Identifying Controlling Features of Engineering Design Iteration", *Management Science*, 43: 276-293.

[74] South Carolina Research Authority (2001), "ISO 10303 STEP Application Handbook Version 2", Viewed August-2003: <www.isg-scra.org/STEP/files/ STEP_Application_Handbook.pdf>.

[75] Steward D.V. (1991), "Planning and Managing the Design of Systems", *Technology Management : the New International Language*: 189 -193.

[76] Stoer M, Wagner F, (1997), "A Simple Min-Cut Algorithm", *Journal of the ACM*, 44(4): 585 – 591.

[77] Suh J.W., Homans S.B., & Yim M. (2002), "Telecubes: Mechanical Design of a Module for Self-Reconfigurable Robotics", *IEEE International Conference on Robotics and Automation 4: 4095 -410.*

[78] Sullivan K.J., Griswold W.G., Cai Y., & Hallen B. (2001), "The Structure and Value of Modularity in Software Design", *ACM SIGSOFT Symposium on the Foundations of Software Engineering, Vienna.*

[79] Tilbury D.M., & Kota S. (1999), "Integrated Machine and Control Design for Reconfigurable Machine Tools", *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Atlanta GA.* Viewed- July 2003: <www-personal.engin.umich.edu/~tilbury/papers/aim99.pdf>.

[80] University of Southern California's Information Sciences Institute (1999), "The USC/ISI Conro Project", Viewed- July 2003: <http://www.isi.edu/conro/>.

[81] Unsal C. , & Khosla P.K. (2000), "Mechatronic Design of a Modular Self-Reconfigurable Robotics System", *Proceedings of the 2000 IEEE International Conference on Intelligent Robots and Systems*, 1742–1747.

[82] Unsal C. (2000), "I-Cubes: A Modular Self-Reconfiguring Bipartite Robotic System", Viewed- July 2003: <http://www-2.cs.cmu.edu/~unsal/research/ices/cubes/>.

[83] Vesanto J., & Alhoniemi E.(2000), "Clustering of the Self-Organizing Map", *IEEE transactions on neural networks, 11(3):* 586 -600.

[84] Vieira F.C., Neto A.D.D., & Costa J.A.F.(2000), "An Efficient Approach to the Traveling Salesman Problem Using Self-Organizing Maps", *International Journal of Neural Systems,* Viewed-August 2003: <www.dee.ufrn.br/alfredo/ijnsVieiraDoriaCosta.pdf>.

[85] Vitanyi P M B, & Li M. (2000), "Minimum Description Length Induction, Bayesianism, and Kolmogorov complexity". *IEEE transactions on Information Theory 46 (2):* 446-464.

[86] Vona M. (1999), "The Crystalline Atomic Unit Modular Self-reconfigurable Robot", Viewed- July 2003: <http://www.ai.mit.edu/~vona/xtal/>.

[87] Walsh, Ronald A. (1998), "Mcgraw-Hill Machining And Metalworking Handbook" 2nd Edition, McGraw-Hill professional, London.

[88] Wikipedia, the free encyclopedia: "Abstract Class", Viewed August'2003 <http://www.wikipedia.org/wiki/Abstract_class>.

[89] Wikipedia, the free encyclopedia: "Class (object-oriented programming)", Viewed August'2003: <http://www.wikipedia.org/wiki/Class_(object-oriented_programming)>.

[90] Wikipedia, the free encyclopedia: "Genetic algorithm", Viewed August'2003: <http://www.wikipedia.org/wiki/Genetic_algorithm>.

[91] Wikipedia, the free encyclopedia: "Thread (computer science)", Viewed August'2003: <http://www.wikipedia.org/wiki/Thread_(computer_programming)>.

[92] Wolfram S. (2002), "A New Kind of Science", Wolfram Media, Inc., Champaign, IL.

[93] Wu Y., Shuming G., & Zichen C. (2001) , "Automatic Setup Planning And Operation Sequencing For Satisfying Tolerance Requirements", *Proceedings of ASME 2001 Design Engineering Technical Conferences, Pittsburgh, PA.*

[94] www.about.com , "Crawler", Viewed August'2003: <http://websearch.about.com/library/glossary/bldef-crawler.htm>.

[95] www.Diffuse.org , "Product Data Representation and Exchange Standards", Viewed August'2003: <http://www.diffuse.org/products.html#Description>.

[96] www.hyperdictionary.com , "Bus" , Viewed August'2003: <http://www.hyperdictionary.com/dictionary/bus>.

[97] www.hyperdictionary.com, "Case Based Reasoning", Viewed August'2003: <www.hyperdictionary.com/computing/case+based+reasoning>.

[98] www.hyperdictionary.com ,"Field-Programmable Gate Array", Viewed August'2003: <http://www.hyperdictionary.com/dictionary/Field-Programmable+Gate+Array>.

[99] www.softwarerecommendations.com, "Feature Creep", Viewed- August 2003: <http://www.softwarerecommendations.com/feature-creep.html>.

[100] Xi F., Ross A., & Lang S. (2001) "Exploring a Re-configurable Parallel Robot for Space Applications", *6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2001, Montreal, Canada.*

[101] Yang G., & Chen I.M (2000), "Task-based Optimization of Modular Robot Configurations: Minimized Degree-of-Freedom Approach" *Mechanism and Machine Theory* 35(4): 517-540.

[102] Yassine A.A., & Browning T.R. (2001), "Analyzing Multiple Product Development Projects Based On Information and Resource Constraints" *Technical report: Ford-MIT alliance.*

[103] Yassine A.A., & Braha D. (2003), "Four Complex Problems in Concurrent Engineering and the Design Structure Matrix Method," *Concurrent Engineering Research & Applications,* 11(3). Sept. 2003.

[104] Yassine A.A., Joglekar N., Braha D., Eppinger S., & Whitney D.(2002), "Information Hiding in Product Development: The Design Churn Effect,", *MIT Sloan School of Management Working Paper-4333-02.*

[105] Yassine A.A., Whitney D.E., Lavine J., & Zambito T.(2000), "Do-it-right-first-time (DRFT) Approach to DSM Restructuring", *Proceedings of ASME 2000 International Design Engineering Technical Conferences September 10-13, 2000, Baltimore, MD.*

[106] Yassine A.A., Whitney D., & Zambito T. (2001), "Assessment of Rework Probabilities for Design Structure Matrix (DSM) Simulation in Product Development Management", *Proceedings of the 13th International Conference on Design Theory and Methodology (DTM 2001).*

[107] Ye B., & Salustri F.A. (2003), "Simultaneous Tolerance Synthesis for Manufacturing and Quality", *Research in Engineering Design,* 14:98-106.

[108] Yim M, Roufas K., Duff D., Zhang Y., & Homans S. (2003), "Modular Reconfigurable Robots in Space Applications", *Autonomous Robot Journal, special issue for Robots in Space,* Springer Verlag, Berlin, Viewed- July 2003: <www2.parc.com/spl/projects/modrobots/ publications/pdf/space.pdf>.

[109] Yu T.L., Yassine A.A., & Goldberg D.E. (2003), "A Genetic Algorithm for Developing Modular Product Architectures," *Proceedings of the ASME 2003 International Design Engineering Technical Conferences, 15th International Conference on Design Theory & Methodology.*

[110] Zhang H C., & Lin E *(1999)*, "A Hybrid-Graph Approach for Automated Setup Planning in CAPP" *Robotics and Computer-Integrated Manufacturing*, 15: 89 –100.

[111] Zhong W. (2002), "Modeling and Optimization for Quality and Productivity for Machining Systems with Different Configurations", *PhD thesis, University of Michigan, Ann Arbor, Michigan.*